Production of solution sheets in LaTeX 2ε

January 16, 2004

Contents

1 Introduction **1**

2 The documentation driver file **1**

3 User interface **2**

4 A simple example **3**

5 A complicated example **3**

6 A further example **5**

7 Identification **6**

8 Options **6**

9 File handling **6**

10 The file interface **7**

11 The solution interface **9**

1 Introduction

This package is a modification of the author's previous style option `answers`, which has been in use for a few years, and was based upon the TeXbook idea of binding solutions to exercises. I have taken the opportunity with this revision to alter the format of the solutions, so that they are now presented as LaTeX environments rather than being started with a command and ended with the end of the surrounding environment, a wholly un-LaTeXy way of doing things!

The other main change is that several file handles are allowed to be active at once. This allows some solutions in a book (for instance) to go to the appendices, and some to go to a separate file, to be printed and handed to the students as the course progresses. Moreover, the actual physical files opened with each file handle can now be varied in the same job, allowing many different files to be created according to the same format. Thus, for instance, each chapter of a book could create its own solution file, allowing the user to use `\include` on both chapters and solutions.

Finally, any number of solution-types may now be bound to any file, not just the two old ones, solution and hint. The format of each solution type is under the complete control of the user.

2 The documentation driver file

This is the driver file that produces this documentation. We use the document class provided by the LaTeX 2_ε distribution for producing the documentation.

```
⟨*driver⟩
\documentclass{ltxdoc}
\RecordChanges
\begin{document}
  \DocInput{answers.dtx}
  \PrintIndex
  \PrintChanges
\end{document}
⟨/driver⟩
```

3 User interface

The package needs to be included with the command

```
\usepackage[nosolutionfiles]{answers}
```

If the optional argument is given, solutions appear at that point in the text, rather than being written to external files. This allows a demonstrator's version to be produced.

`\Newassociation` After that, there should be several declarations of the form

```
\Newassociation{xxx}{yyy}{zzz}
```

where `xxx` is an environment in the document, and `yyy` is an environment which will surround the contents of `xxx` when it is written to symbolic file handle `zzz`. The names `xxx`, `yyy` and `zzz` should consist of letters only, not numbers, punctuation or spaces.

`\solutionextension` By default, output will go to `zzz.tex` if `zzz` is open. The command `\solutionextension` can be redefined to change `tex` to some other extension. Alternatively, the output filename can be changed as an optional parameter to `\Opensolutionfile`, and each `\Opensolutionfile` on the same handle can use a different physical file. By default, `\solutionpoint` is added before `\solutionextension`. Redefine it to remove it. (It has the obvious default value of a period.

`\Opensolutionfile` `\Closesolutionfile` At some point the user types

```
\Opensolutionfile{zzz}
...
\Closesolutionfile{zzz}
```

to create a file of solutions written by environments `xxx` to environments `yyy`. If this construction is used several times, then several files of solutions will be created. The user may wish these files to have different names. If the form `\Opensolutionfile{zzz}[www]`, then `www.tex` is used as actual file output name rather than `zzz.tex`. This allows file handle `zzz` to create many files `www.tex`, say one for each chapter of a book, or one for each problem sheet. These could then be processed using `\include` commands. The same value of `\solutionextension` is used for the optional argument as for the main argument. The name `www` should follow the usual file naming conventions.

`\Writetofile` In addition, material can be written directly to a file by means of `\Writetofile`. Its first argument is the file handle `zzz` and its second is the line of text to be written. It is most important to remember that any control words in the line to be written should be preceded by `\protect`, otherwise the primitive TEX `\write` command will expand them. Also, as the argument is read in TEX' usual way before being written, any trailing spaces after a control word will disappear unless precautions are taken. Thus, to write `\xx yyy` to the file, the user can type `\protect\xx\space yyy`.

`Filesave` Alternatively, a block of text can be saved to file handle `zzz` by means of

```
\begin{Filesave}{zzz}
   ....
\end{Filesave}
```

around it once, `zzz` has been opened. The restrictions that apply to `\Writetofile` above do not apply to this environment.

`\Readsolutionfile` One of the generated files can be read using

```
\Readsolutionfile{zzz}
```

provided the file has not been closed and re-opened. Alternatively, simply `\input` or `\include` it if preferred.

None of the file operations should have any effect if the file handle `zzz` has not been opened, or if `nosolutionfiles` is specified.

4 A simple example

Here is a straightforward example to illustrate how these macros are used.

```
⟨*ex1⟩
\documentclass[12pt,a4paper]{article}
\usepackage{answers}
\Newassociation{sol}{Solution}{ans}
\newtheorem{ex}{Exercise}
\begin{document}
\Opensolutionfile{ans}[ans1]
\section{Problems}
\begin{ex}
   First exercise
   \begin{sol}
      First solution.
   \end{sol}
```

```
\end{ex}
\begin{ex}
   Second exercise
   \begin{sol}
      Second solution.
   \end{sol}
\end{ex}
\Closesolutionfile{ans}
\section{Solutions}
\input{ans1}
\end{document}
```
⟨/ex1⟩

5 A complicated example

The following is an (over-complicated) example of the use of package `answers`. It uses some of the refinements described later.

⟨*ex2⟩
```
\documentclass[12pt,a4paper]{article}
\usepackage{answers}%\usepackage[nosolutionfiles]{answers}
```

First an environment which contains problems and numbers them. This is based on a LaTeX theorem, but with a roman body rather than italic.

```
\newtheorem{Exc}{Exercise}
\newenvironment{Ex}{\begin{Exc}\normalfont}{\end{Exc}}
```

Three sorts of solution are written to two different files. File handle `test` will contain the solutions and hints that the students will see; `testtwo` contains the solutions to the problems which they will probably hand in, and so these must be formatted separately.

```
\Newassociation{solution}{Soln}{test}
\Newassociation{hint}{Hint}{test}
\Newassociation{Solution}{sSol}{testtwo}
```

Because we want to mark different types of problem in the master file of problems, we define the following.

```
\newcommand{\prehint}{~[Hint]}
\newcommand{\presolution}{~[Solution]}
\newcommand{\preSolution}{~[Homework]}
```

We provide an extra parameter when we open file handle `test`; this is because we want to write a `\section` command to the solution file. This is merely an illustration here, but would be more relevant if the solution file were `\included`.

```
\newcommand{\Opentesthook}[2]%
   {\Writetofile{#1}{\protect\section{#1: #2}}}
```

The default text produced when LaTeXmeets the solution environments is here modified.

```
\renewcommand{\Solnlabel}[1]{\emph{Solution #1}}
\renewcommand{\Hintlabel}[1]{\emph{Hint #1}}
\renewcommand{\sSollabel}[1]{\emph{Solution to #1}}

\begin{document}
```

We open handle `test` as actual file `test1.tex`,

```
\Opensolutionfile{test}[ans2]{Solutions}
```

and write some text on it.

```
\Writetofile{test}{\protect\subsection{Some Solutions}}
```

Handle `testtwo` is opened as `testtwo.tex`.

```
\Opensolutionfile{testtwo}[ans2x]
\Writetofile{testtwo}{%
   \protect\subsection{Extra Solutions}}
```

Now the problems.

```
\section{Exercises}
\begin{Ex}
   An exercise with a solution.
   \begin{solution}
      This is a solution.
      \relax{}
   \end{solution}
\end{Ex}
\begin{Ex}
   An exercise with a hint and a secret solution.
   \begin{hint}
      This is a hint.
   \end{hint}
   \begin{Solution}
      This is a secret solution.
   \end{Solution}
\end{Ex}
\begin{Ex}
   An exercise with a hint.
   \begin{hint}
      This is a hint.
   \end{hint}
\end{Ex}
```

We close the two solution files and immediately input their contents. We could have used `\include` here.

```
\Closesolutionfile{test}
\Readsolutionfile{test}
\clearpage
\Closesolutionfile{testtwo}
\Readsolutionfile{testtwo}
\end{document}
⟨/ex2⟩
```

6 A further example

Here is an application to a situation not originally envisaged, suggested to the author by Martin Osborne. Here, the exercises and solutions are not numbered; they are *described*.

```
⟨*ex3⟩
\documentclass[12pt,a4paper]{article}
\usepackage{answers}
```

```
\newenvironment{Ex}[1]{\begin{trivlist}\item \emph{#1} %
   \renewcommand{\Currentlabel}{#1}}{\end{trivlist}}
\Newassociation{solution}{Soln}{solutions}

\renewenvironment{Soln}[1]{\begin{trivlist}\item
   Solution to \emph{#1} }{\end{trivlist}}

\begin{document}
\section*{Problems}
   \Opensolutionfile{solutions}[ans3]
   \begin{Ex}{First exercise}
      An exercise with a solution.
      \begin{solution}
         This is a solution.
         \relax{}
      \end{solution}
   \end{Ex}
   \begin{Ex}{Second exercise}
      A second exercise with a solution.
      \begin{solution}
         This is another solution.
      \end{solution}
   \end{Ex}
   \Closesolutionfile{solutions}
\section*{Solutions}
   \Readsolutionfile{solutions}
\end{document}
```
⟨/ex3⟩

7 Identification

This package can only be used with LaTeX 2_ε, so an appropriate message is displayed when another format is used.

⟨∗answers⟩
```
\NeedsTeXFormat{LaTeX2e}[1996/06/01]
```

Announce the package name and its version:

```
\ProvidesPackage{answers}[1996/07/10]
```

And display it on the terminal (and the log file):

```
\typeout{Package `answers' <\filedate>.}
\typeout{\Copyright}
```

8 Options

There is a single option `nosolutionfiles` that switches output off to files and produces the solutions here-and-now.

```
\newif\ifanswerfiles \answerfilestrue
\DeclareOption{nosolutionfiles}{\answerfilesfalse
   \typeout{No answer files being produced}}%
\ProcessOptions
```

As this package now relies heavily on the `verbatim` package, we ensure that that is loaded.

```
\RequirePackage{verbatim}
```

9 File handling

`\solutionextension` The default extension for solution files is defined here.

```
\newcommand{\solutionpoint}{.}
\newcommand{\solutionextension}{tex}
```

It may be changed with `\renewcommand`.

`Filesave` We define an environment `Filesave` with one parameter, the file handle. It is similar to the example of Schöpf in the description of `verbatim`.

```
\newenvironment{Filesave}[1]{%
   \@bsphack
   \def\verbatim@processline{}%
   \Iffileundefined{#1}{}{%
      \Ifopen{#1}{%
         \def\verbatim@processline{%
            \Ifanswerfiles{%
               \immediate\write\@nameuse{#1@file}%
                  {\the\verbatim@line}%
            }{}%
         }%
      }{}%
   }%
   \let\do\@makeother\dospecials
   \catcode`\^^M\active \catcode`\^^I=12\relax
   \verbatim@start
}{\@esphack}
```

`\Writetofile` It is also useful to have a command to write material to the file. In this, you need to put `\protect` before any control words in the argument that might expand prematurely and create havoc.

```
\newcommand{\Writetofile}[2]{%
   \@bsphack
   \Iffileundefined{#1}{}{%
      \Ifopen{#1}{%
         {%
            \let\protect\string
            \Ifanswerfiles{%
               \immediate\write\@nameuse{#1@file}{#2}%
            }{}%
         }%
      }{}%
   }%
   \@esphack
}
```

`\Ifopen` We need to check whether or not a file is already open.

```
\newcommand{\Ifopen}[3]{%
   \csname if#1open\endcsname#2\else#3\fi}%
```

\Iffileundefined We also need to check whether a file variable has already been defined for a given file handle.

```
\newcommand{\Iffileundefined}[3]{%
   \csname ifx\expandafter\endcsname
      \csname #1@file\endcsname\relax
      #2\else#3\fi}
```

Finally, we need a check as to whether we are outputting answers to a file or not

```
\newcommand{\Ifanswerfiles}[2]{%
   \ifanswerfiles #1\else #2\fi}
```

10 The file interface

\Opensolutionfile Before we can write solutions, we must open the solution file(s). The command to do this takes a single parameter, which should usually be a file name without extension. Thus it should probably be restricted to a string of at most 8 letters for portability. This operation will not truncate any existing open file. However, if the second optional parameter is specified, this determines the actual filename, and the first parameter is then an arbitrary symbolic file handle name.

```
\def\Opensolutionfile#1{%
   \@ifnextchar[{\define@filename{#1}}%
      {\define@filename{#1}[#1]}}%
\def\define@filename#1[#2]{%
   \global\@namedef{#1@filename}{#2\solutionpoint\solutionextension}%
   \Ifanswerfiles{%
      \typeout{Output from handle #1 going
         to #2.\solutionextension}%
   }{}%
   \Iffileundefined{#1}{%
      \expandafter\newwrite\csname #1@file\endcsname
      \csname newif\expandafter\endcsname
         \csname if#1open\endcsname
      \global\csname #1openfalse\endcsname
      \expandafter\ifx\csname Open#1hook\endcsname\relax
         \global\@namedef{Open#1hook}##1{}%
      \fi
      \expandafter\ifx\csname Close#1hook\endcsname\relax
         \global\@namedef{Close#1hook}##1{}%
      \fi
   }{}%
   \let\Tmp\relax
   \Ifopen{#1}{\typeout{File #1 already open}}{%
      \Ifanswerfiles{%
         \immediate\openout\@nameuse{#1@file}=%
         \@nameuse{#1@filename}%
      }{}%
      \global\csname#1opentrue\endcsname
      \def\Tmp{\@nameuse{Open#1hook}{#1}}%
   }%
   \Tmp
}
```

\Closesolutionfile We also have a command to close an already open file.

```
\def\Closesolutionfile#1{%
   \let\Tmp\relax
   \Iffileundefined{#1}{}{%
      \Ifopen{#1}{%
         \Ifanswerfiles{%
            \immediate\closeout\@nameuse{#1@file}%
         }{}%
         \global\csname #1openfalse\endcsname
         \def\Tmp{\@nameuse{Close#1hook}{#1}}%
      }{}%
   }%
   \Tmp
}
```

Note that the two file commands each provide a hook which allows them to perform extra tasks. For instance, the opening operation could be made to write extra information to the file by redefining the appropriate hook. The closing operation could if required do an immediate `\input` of the solution file contents. For example,

```
\newcommand{\Openxxxhook}[2]{%
   \Writetofile{#1}{\protect\section{#2}}%
}%
\newcommand{\Closexxxhook}[1]{%
   \Readsolutionfile{#1}%
}
```

and then

```
\Opensolutionfile{xxx}{Answers to selected problems}
...
\Closesolutionfile{xxx}
```

The default behaviour is to ignore the one parameter. Note that if you redefine their behaviour, you must remember that the first parameter is always the file handle.

\Readsolutionfile The operation of reading the file of solutions can be done with the following command.

```
\def\Readsolutionfile#1{%
   \Ifanswerfiles{%
      \Iffileundefined{#1}{}{%
         \Ifopen{#1}{%
            \typeout{WARNING: attempt to read open file #1}%
         }{%
            \edef\Tmp{%
               \noexpand\InputIfFileExists
                  {\@nameuse{#1@filename}}{}%
               {\noexpand\message{File
                  \@nameuse{#1@filename}%
                     \space not found}}%
            }%
```

```
            \Tmp
         }%
      }%
   }{}%
}
```

11 The solution interface

`\Newassociation` Several solution file handles may have been defined. You are limited only by the number that TEX will make available to you. Each solution environment that is to write to one of these handles must know which handle to write to, and also what extra information to write there, apart from its contents. This is done by setting up an association between the source environment, the destination environment and the file handle.

```
\newcommand{\Newassociation}[3]{%
   \newsolution{#2}%
   \expandafter\ifx\csname #3opentrue\endcsname\relax
      \expandafter\newif\csname if#3open\endcsname
   \fi
   \newenvironment{#1}{%
      \Ifanswerfiles{%
         \let\Tmp\relax
         \Iffileundefined{#3}{}{%
            \Ifopen{#3}{%
               \immediate\write\@nameuse{#3@file}%
                  {\string\begin{#2}\@nameuse{#2params}}%
            \def\Tmp{\Filesave{#3}}%
            }{}%
         }%
      }{%
         \edef\Tmp{\noexpand\begin{#2}\@nameuse{#2params}}%
      }%
      \csname pre#1\endcsname
      \Tmp
   }%
   {%
      \Ifanswerfiles{%
         \Iffileundefined{#3}{}{%
            \Ifopen{#3}{%
               \endFilesave%
               \immediate\write\@nameuse{#3@file}%
                  {\string\end{#2}}%
               \csname post#1\endcsname
            }{}%
         }%
      }{%
         \end{#2}%
      }%
   }%
}
```

`\newsolution` The default destination environment in the solution file is defined to take a single

parameter, a reference number inherited from the source environment. This is set with style `\solutionstyle`, which defaults to `\textbf`. In addition, solution type `yyy` can have markup added before and after it by defining `\preyyy` and `\postyyy` suitably, eg, a rule across the width of the page or a square. If anything more sophisticated is intended, it is probably better to `\renewenvironment{yyy}` to achieve it.

`\Currentlabel`

```
\newcommand{\newsolution}[1]{%
   \@ifundefined{#1}{%
      \global\@namedef{#1params}{{\Currentlabel}}%
      \newenvironment{#1}[1]%
      {%
         \csname pre#1\endcsname
         \trivlist
         \item[\hskip\itemsep{\@nameuse{#1label}{##1}}]}%
      {\csname post#1\endcsname\endtrivlist}%
      \global\@namedef{#1label}##1{\solutionstyle{##1}}%
   }{\typeout{WARNING: environment #1 already in use}}%
}
\newcommand{\solutionstyle}[1]{\textbf{#1}}
\newcommand{\Currentlabel}{\@currentlabel}
```

The format of the label for solution environment `xxx` is governed by the command `\xxxlabel`, which takes one argument by default. The argument is passed to it by the command `\xxxparams`, which expands to `{\Currentlabel}`, a synonym for `{\@currentlabel}`, and this argument is written automatically by the source environment. The label appears in boldface by default. We could easily change the behaviour of this environment by changing these two commands. For example

```
\renewcommand{\xxxlabel}[1]{\emph{Solution to #1}}
\renewcommand{\xxxparams}{{\Currentlabel(p.\thepage)}}
```

would provide a number and page reference in italic.

More complicated behaviour could be produced by redefining the `xxx` environment itself to take a different number of parameters. Note however that `\xxxparams` must be redefined to provide those parameters.

Which brings us to the end of the `answers` package.

⟨/answers⟩