mi-solns: Extract solutions from exercises and quizzes

D. P. Story
Email: dpstory@uakron.edu

processed January 9, 2019

Contents

1 ⟨∗package⟩

1 Introduction

The exerquiz package is capable of creating questions and solutions to exercises and quizzes. The purpose of this package is to develop commands for extracting any desired solution, for whatever reason, and typesetting it anywhere in the body of the document. To accomplish this goal, the recent version of exerquiz is required (dated 2018/12/13 or later) and the shellesc package.

While the document is being compiled, the solution files (SOL and QSL) are being written to, so we cannot input them or read them. What we do is to make a copy of the solution files from within the operating system, and input that back into the body of the document when required. Consequently, it is necessary to activate the feature of executing an OS script from within the compiling

operation. To activate the feature, the document needs to be compiled with the `--shell-escape` switch (for latex, pdflatex, lualatex, and xelatex).

The basic idea is to mark a solution that is to be reproduced elsewhere, `\mrkForIns{`⟨*name*⟩`}`, just above the `solution` environment. Then elsewhere in the document, input the solution using the command `\insExSoln{`⟨*name*⟩`}`, `\insSqSoln{`⟨*name*⟩`}`, or `\insQzSoln{`⟨*name*⟩`}`, depending on whether the solution came from the `exercise`, `shortquiz`, or `quiz` environment.

2 Preliminaries

```
\RequirePackage{shellesc}
\ProcessOptions\relax
```

We require either exerquiz or eqexam with a minimal publish date, `\mi@reqChk` performs the check, but delays it until `\AtBeginDocument`.

```
\def\mi@reqChk{\begingroup\mifoundfalse
  \@ifpackageloaded{exerquiz}{\mifoundtrue\def\reqDate{2018/12/13}
    \@ifpackagelater{exerquiz}{\reqDate}
      {}{\PackageWarning{mi-solns}{exerquiz dated \reqDate\space
        or later\MessageBreak required}}%
  }{}%
  \@ifpackageloaded{eqexam}{\mifoundtrue
      \def\reqDate{2018/12/13}\@ifpackagelater{eqexam}{\reqDate}
      {\mi@solutionsonlyfix}{\PackageWarning{mi-solns}
        {eqexam dated \reqDate\space or\MessageBreak
         later required}}%
  }{}%
  \ifmifound\else
    \PackageWarning{mi-solns}{For this package to be
    effective\MessageBreak you need either exerquiz
    or eqexam, as appropriate}\fi
\endgroup}
\AtBeginDocument{\mi@reqChk}
```

Beginning with exerquiz/eqexam dated 2018/12/13, we can change the name of the SOL file (from its default of `\jobname.sol`. If the `solutionsonly` option of eqexam is being applied, we made some changes so that option works with mi-solns.

```
\def\mi@solutionsonlyfix{\ifsolutionsonly
  \edef\eqExSolFileName{\misolout}\expandafter
  \global\copySolnsOff\global\notamiopfalse\fi}
```

3 Core commands for this package

`\copyfileCmdEx`
`\copyfileCmdQz`

The commands to copy SOL and QSL files, may be redefined as needed in the operating system.

```
\def\declSOLIn#1{\def\misolin{#1}}\def\declSOLOut#1{\def\misolout{#1}}
\def\misolin{\jobname.sol}\def\misolout{\jobname-cpy.sol}
\def\declQSLIn#1{\def\miqslin{#1}}\def\declQSLOut#1{\def\miqslout{#1}}
\def\miqslin{\jobname.qsl}\def\miqslout{\jobname-cpy.qsl}
```

```
\newcommand*{\copyfileCmdEx}{copy \misolin\space\misolout}
\newcommand*{\copyfileCmdQz}{copy \miqslin\space\miqslout}
\def\mi@copysolns{%
  \ShellEscape{\copyfileCmdEx}\ShellEscape{\copyfileCmdQz}}
```

`\copySolnsOn` Making a copy of the solution files is the default.

`\copySolnsOff` After there are no more changes to the solution files, you can say `\copySolnsOff`, and each compile will not rebuild '`cpy-`' solution files.

```
\def\copySolnsOn{\let\mi@copySolns\mi@copysolns}
\def\copySolnsOff{\let\mi@copySolns\relax}
\@onlypreamble\copySolnsOn
\@onlypreamble\copySolnsOff
\copySolnsOn
```

Copy the solution files. At the end of the document, we make a copy of the solution files, provided `\copySolnsOn` is in effect.

```
\AtEndDocument{\mi@copySolns}
```

Some utility commands Below, we define a few useful commands.

`\ignoreterminex` `\ignoreques` If `\eqterminex` has a special definition, perhaps created from the `cq` environment, you can pass `\ignoreterminex` through the optional argument of `\insExSoln` and the question to the problem does not appear. We also declare `\ignoreques` as an alias for `\ignoreterminex`.

`\ifmifound` `\ifnotamiop` We define two switches: `\ifmifound` is set to true when a ⟨*name*⟩ we are searching for is found; otherwise it remains false. A value of false causes a warning to be issued. `\ifnotamiop` (not a MI operation) is used to control what is displayed when one of the `\ins...` commands are used. See comments just below.

```
\newif\ifmifound \mifoundfalse
\newif\ifnotamiop \notamioptrue
\newif\ifmi@OKtoRead \mi@OKtoReadtrue
\def\readSolnsOn{\mi@OKtoReadtrue}
\def\readSolnsOff{\mi@OKtoReadfalse}
\newcommand*{\miReadOffMsg}{(\textbf{?? read is off ??})}
```

`\writeToExSolns` `\writeToSolnFile` `\writeToQzSolns` The document author can can write to the solution files using `\writeToExSolns`, `\writeToSolnFile`, or . The macros are originally defined in exerquiz and eqexam, but we redefine them here so their argument is enclosed in `\ifnotamiop`.

```
\newcommand\mi@wrt@fix[1]{\protect\ifnotamiop^^J%
  #1^^J\protect\fi}
\renewcommand\writeToExSolns[1]{\writeT@ExSolns{\mi@wrt@fix{#1}}}
\renewcommand\writeToQzSolns[1]{\writeT@QzSolns{\mi@wrt@fix{#1}}}
\@ifpackageloaded{eqexam}
  {\renewcommand\writeToSolnFile[1]{\writeT@SolnFile{\mi@wrt@fix{#1}}}}
  {\let\writeToSolnFile\writeToExSolns}
\def\ignoreterminex{\let\eqterminex\relax\let\decleqterminex\@gobble}
\let\ignoreques\ignoreterminex
```

Some gobbling macros.

```
\long\def\gobbleiiterminex#1\eqterminex{}
\long\def\gobbleiiendinput#1\endinput{\endinput}
\long\def\gobbleiiendgroup#1\endgroup{}
\long\def\mi@griii#1#2#3{}
```

`\mi@griii` `\eqgriii` is eventually `\let` to`\mi@griii`, while `\eqgrii` is `\let` to `\@gobbletwo`. In the data structure of the solution file of an eqexam document. In such a document, `\eqgriii` and `\eqgrii` appears at the top and bottom of the file; for example,

```
\eqgriii\noindent\begin{eqequestions}
...
\eqgrii\end{eqequestions}
```

We don't want the `eqequestions` environment input as part of the insertion, so we must gobble them up using `\eqgriii` (`\let` to `\mi@griii`) and `\eqgrii` (`\let` to `\@gobbletwo`). These are formatting (a list env), which we don't want in the body of our document. We are just trying to input the ⟨*solution*⟩ part of the data structure; everything else needs to be ignored.

The next three commands do some internal work. They are each called by `\insExSoln`, `\insSqSoln`, and `\insQzSoln`, respectively

3.1 Internal macros that expand within a solutions file

There are three macros that are defined and are executed as a solution file is input.

3.1.1 For exercises

Exercises are the more difficult case because they are used not only by exerquiz, but also eqexam; in the latter, there are may more 'control' commands that are written to the solution file (SOL) to format how the solutions appear at the end of the document.

Below is a representation of a solution to an exercise. The `\insExSoln` command, `\lets` `\eqMrkSoln` to `\eqMrkSolnCpyEx`

A representative data structure for an exercise (exerquiz)

```
\eqMrkSoln{⟨name⟩}\eqEXt{}{}\solnItemMngt
\exerSolnHeader{⟨argi⟩}{⟨argii⟩}{argiii}\eqterminex
    ⟨solution⟩
\ReturnTo{⟨argi⟩}{⟨argii⟩}\endeqEXt\par{\medskip}%
```

If a solution to an exercise (or quiz) is *not marked* by `\mrkForIns`⟨*name*⟩, then `\eqMrkSoln{`⟨*name*⟩`}` *does not appear* in the structure.

A representative data structure for an problem (eforms)

```
\decleqterminex{\cqFmtPasteQues{cq-1.cut}}%
\eqMrkSoln{⟨name⟩}\eqEXt{}{}\solnItemMngt
\exerSolnHeader{⟨argi⟩}{⟨argii⟩}{⟨argiii⟩}\selectVersion{}{3}\eqterminex
    ⟨solution⟩
\ReturnTo{⟨argi⟩}{⟨argii⟩}\endeqEXt\par{{}}%
```

When `\selectVersion` *does not appear* prior to the `solution` environment, then the `\declareterminex` and `\selectVersion` (and args) *do not appear* in the structure.

In general. When `\eqMrkSoln` is expanded, it tests whether ⟨*name*⟩ matches `\eqMrkCpyArg`. Referencing the above data structures, `\insExSoln` pretty much sets `\eqEXt`, `\solnItemMngt`, `\exerSolnHeader`, `\ReturnTo` to gobble their argument and become noops. `\endeqEXt` gobbles everything up down to `\endinput`.

`\eqMrkSolnCpyEx{`⟨*name*⟩`}` Process a marked solution for an exercise. We hunt for ⟨*name*⟩.

```
\def\eqMrkSolnCpyEx#1{\def\eqargi{#1}%
  \ifx\eqargi\eqMrkCpyArg
    \mifoundtrue
    \let\par\par@SAVE
    \ifmakeExSlLocal
      \long\def\endeqEXt##1##2##3{##3\gobbleiiendinput}\else
      \let\endeqEXt\gobbleiiendinput\fi
      \let\eqEXt\@gobbletwo
      \let\mi@next\relax
  \else
    \long\def\endeqEXt##1##2{}%
    \let\mi@next\gobbleToEndEXt
  \fi
\mi@next}
```

3.1.2 For short-quizzes

A representative data structure for a short-quiz (exerquiz)

```
\eqMrkSoln{⟨name⟩}\eqSQt{}\quizSolnHeader{⟨argi⟩}{⟨argii⟩}\eqterminex
    ⟨solution⟩
\ReturnTo{⟨argi⟩}{⟨argii⟩}\endeqQt\fpAfterSolutionsSkip
```

As commented above, `\eqMrkSoln` may not appear in the data structure.

`\eqMrkSolnCpySQ{`⟨*name*⟩`}` Process a marked solution for an short-quiz (`shortquiz` env). We hunt for ⟨*name*⟩.

```
\def\eqMrkSolnCpySQ#1{\def\eqargi{#1}%
  \ifx\eqargi\eqMrkCpyArg
    \mifoundtrue
    \let\par\par@SAVE
    \ifmakeQzSlLocal
      \long\def\endeqSQt##1##2{##2\gobbleiiendinput}\else
      \let\endeqSQt\gobbleiiendinput
    \fi
```

```
    \let\mi@next\gobbleiiterminex
  \else
    \long\def\endeqSQt##1{}%
    \let\mi@next\gobbleToEndSQt
  \fi
\mi@next}
```

3.1.3 For quizzes

A representative data structure for a quiz (exerquiz)

```
\eqMrkSoln{⟨name⟩}\eqQt{}\quizSolnHeader{⟨argi⟩}{⟨argii⟩}\eqterminex
    ⟨solution⟩
\ReturnTo{⟨argi⟩}{⟨argii⟩}\endeqQt\fpAfterSolutionsSkip
```

As commented above, `\eqMrkSoln` may not appear in the data structure.

`\eqMrkSolnCpyQz{`⟨*name*⟩`}` Process a marked solution for an quiz (`quiz` env). We hunt for ⟨*name*⟩.

```
\def\eqMrkSolnCpyQz#1{\def\eqargi{#1}%
  \ifx\eqargi\eqMrkCpyArg
    \mifoundtrue
    \let\par\par@SAVE
    \ifmakeQzSlLocal
      \long\def\endeqQt##1##2{##2\gobbleiiendinput}\else
      \let\endeqQt\gobbleiiendinput
    \fi
    \let\mi@next\gobbleiiterminex
  \else
    \long\def\endeqQt##1{}%
    \let\mi@next\gobbleToEndSQt
  \fi
\mi@next}
```

4 User commands for inserting a solution

The main commands are `\insExSoln`, `\insSqSoln`, and `\insQzSoln`.

Preliminary to the definitions, we define a command that is common to all of them. When inputting a solutions file, we cannot any vertical spaces that are not part of the solution, not formatting, not list environments, and so on. The `\mi@nullify` is designed to cancel, nullify, or otherwise neutralize anything that is unwanted. I've added `\addToMINullify` for any unforseen things we don't want to appear or to affect spacing.

`\mi@nullify`
`\addToMINullify`

```
\let\addToMINullify\relax
\def\mi@nullify{\let\par@SAVE\par\let\par\relax
  \let\eqgrii\@gobbletwo\let\eqgriii\mi@griii\let\solnItemMngt\relax
  \def\exerSolnHeader##1##2##3{}\def\ReturnTo##1##2{\unskip}%
  \let\eqTopOfSolnPage\relax\let\preExamSolnHead\relax
  \let\eqTopOfQslPage\relax
  \let\examSolnHeadFmt\@gobble\let\postExamSolnHead\relax
  \let\btwnExamSkip\relax\def\quizSolnHeader##1##2{}\addToMINullify}
```

4.1 For exercises

`\insExSoln[`⟨*inserts*⟩`]{`⟨*name*⟩`}` Used for displaying the solution to an exercise that has been marked by `\mrkForIns{`⟨*name*⟩`}`. The optional argument (⟨*inserts*⟩) is passed (inserted) into the top of the `\insExSoln`. For exercises, there is a `cq` command that copies the question to the solution. By default, the question *is displayed*; however, by passing `\ignoreterminex` the question *is not displayed*.

`\ignoreterminex`

```
\newcommand{\insExSoln}[2][]{\begingroup\withinsoldoctrue#1\relax
  \notamiopfalse\mi@nullify
  \let\eqMrkSoln\eqMrkSolnCpyEx
```

Setting `\useExtFilter` and `\filterFor{@NOMATCH@}`; hopefully, no match is ever obtained, which means the already existent function of `\useExtFilter` will skip over the entries. When `\eqMrkSoln` does not appear in a data structure, the filter will care of the structure entry.

```
  \useEXtFilter\filterFor{@NOMATCH@}\def\eqMrkCpyArg{#2}%
  \ifmi@OKtoRead\InputIfFileExists{\misolout}{}{}\ifmifound\else
  \textbf{??}\PackageWarning{mi-solns}{The name '#2' defined by
    \string\mrkForIns\MessageBreak was not found}\fi\else
    \miReadOffMsg\fi\endgroup}
```

4.2 For quizzes

The commands `\insSqSoln` and `\insQzSoln` are identical, except for two parameters: `#3` is the internal command (`\eqMrkSolnCpySQ` or `\eqMrkSolnCpyQz`); while `#4` is the normal filter (`\useSQtFilter` or `\useQtFilter`).

```
\newcommand\mi@insSQzSoln[4][]{\begingroup\withinqsldoctrue#1\relax
  \notamiopfalse\mi@nullify
  \let\eqMrkSoln#3\relax
  #4\filterFor{@NOMATCH@}\def\eqMrkCpyArg{#2}%
  \ifmi@OKtoRead\InputIfFileExists{\miqslout}{}{}\ifmifound\else
  \textbf{??}\PackageWarning{mi-solns}
  {The name '#2' defined by \string\mrkForIns\MessageBreak
    was not found}\fi\else\miReadOffMsg\fi\endgroup}
```

`\insSqSoln[`⟨*inserts*⟩`]{`⟨*name*⟩`}` Used for displaying the solution to an short-quiz that has been marked by `\mrkForIns{`⟨*name*⟩`}`. The optional argument (⟨*inserts*⟩) is passed (inserted) into the top of the `\insSqSoln`.

```
\newcommand{\insSqSoln}[2][]{%
  \mi@insSQzSoln[#1]{#2}{\eqMrkSolnCpySQ}{\useSQtFilter}}
```

`\insQzSoln[`⟨*inserts*⟩`]{`⟨*name*⟩`}` Used for displaying the solution to an quiz that has been marked by `\mrkForIns{`⟨*name*⟩`}`. The optional argument (⟨*inserts*⟩) is passed (inserted) into the top of the `\insQzSoln`.

```
\newcommand{\insQzSoln}[2][]{%
  \mi@insSQzSoln[#1]{#2}{\eqMrkSolnCpyQz}{\useQtFilter}}
```

⟨/package⟩

5 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.

6 Change History

v0.6 (2018/12/28)

General: Change package name from ci-solns to mi-solns; change internal commands to reflect this renaming 2