The filecontentsdef package

Jean-François Burnol
jfbu (at) free (dot) fr

Package version: v1.2 (2016/09/19); documentation date: 2016/09/19.
From source file filecontentsdef.dtx. Time-stamp: <19-09-2016 at 11:46:39 CEST>.

Abstract

This lightweight LaTeX2e package provides environments `filecontentsdef` and `filecontentshere`. They are derived from the LaTeX `filecontents` environment as extended by Scott Pakin's filecontents package.[1] In addition to the file creation they either store the (verbatim) contents in a macro (`filecontentsdef`) or typeset them (verbatim) on the spot (`filecontentshere`).

I developed this to display TeX code verbatim in documentation and simultaneously produce during the LaTeX run the corresponding files in order to embed them in the PDF as *file attachment annotations* (via the services of Scott Pakin's further package attachfile.)

1 Description

filecontentshere

The environment

```
\begin{filecontentshere}{<filename>}
... arbitrary contents ...
\end{filecontentshere}
```

creates on the fly a file with these contents, and simultaneously it typesets them in a verbatim environment. There is no syntax highlighting whatsoever.

1. The contents are not completely arbitrary, as they may not contain `\end{filecontentshere}` itself...
2. This uses underneath the `verbatim` environment and this has been tested to be compatible with the standard `verbatim`, with the one from package doc (classes `ltxdoc.cls`, `scrdoc.cls`) and also with the one from package verbatim (whose mechanism is quite different from the one of the default `verbatim` environment.)

filecontentsdef

The other environment is `filecontentsdef`. It has a second mandatory argument, a macro.

```
\begin{filecontentsdef}{<filename>}{\macro}
... arbitrary contents ...
\end{filecontentsdef}
```

It creates the file and rather than typesetting it verbatim simultaneously, it stores its (verbatim) contents in its second argument `\macro`.

1. the scope of the macro definition is global,
2. `filecontentshere` is a wrapper of the `filecontentsdef` environment using `\filecontentsheremacro` as the macro where the contents are stored. This macro can then be reused elsewhere if wanted.

\filecontentsheremacro

[1] `filecontentsdef` works independently from filecontents and does not load it.

3. both environments admit the starred form which does *not* add the usual three comment lines at the top of the written file (those lines are anyhow not typeset by `filecontentshere` nor are they included in the macro by `filecontentsdef`).

> Please note that `filecontentsdef` basically stores sanitized (i.e. verbatim) tokens in its macro argument `\macro`.
>
> If the material consists of LaTeX code, the expansion of the macro will only typeset some *verbatim* rendering of the LaTeX code.

Using ε-TeX's `\scantokens`, one can re-tokenize the macro contents and (here naturally, we are talking about the situation where the macro contains TeX/LaTeX code): we obtain its "execution" via `\scantokens\expandafter{\macro}`. Due to the way `\scantokens` works, this must be done with `\newlinechar` set to 13 (to match the `^^M`'s; see later in this documentation). Example:

```
\begin{filecontentsdef}{\jobname.test}{\macro}
  \begin{framed}
    \noindent
                We have coded this in \LaTeX: both
      $E=mc^2$ (input as |$E=mc^2$|)
    and     $E=h\nu$  owe much    to \textsc{Albert Einstein}.
  \end{framed}
\end{filecontentsdef}
{\newlinechar13 \scantokens\expandafter{\macro}}
```

> We have coded this in LaTeX: both $E = mc^2$ (input as `$E=mc^2$`) and $E = h\nu$ owe much to Albert Einstein.

Notice[2] that a space token will generally appear at the end of the expansion, due to `\scantokens`'s way of working. This is an end-of-line space, which we could suppress via `\endlinechar-1\relax` before the `\scantokens`, but that is an option only in the case of single-line contents. If we had written above `\end{framed}%` or `\end{framed}\relax` in our use of `filecontentsdef` this would have prevented `\scantokens` from inserting this final space (naturally in this example the space is issued while TeX is in vertical mode and leaves no trace anyhow).[3]

`\filecontentsexec\macro`

Although `filecontentsdef` itself makes no use of ε-TeX, it provides as a convenience `\filecontentsexec` which will use `\scantokens` to execute a `\macro` as above (thus assuming it contains legitimate but possibly verbatimized LaTeX code.) Rather than using a group (possibly `\macro` makes some non-global definitions) it issues `\newlinechar10\relax` (as this is the default – we could have stored and restored current value, but well...) after the `\scantokens`.

As an example consider the following (with some `utf8` characters among those which are available in `T1`-encoded TeX-fonts as used by this document with the help of `fontenc` and `inputenc`):

```
\begin{filecontentsdef}{filecontentsdef.license}{\fcdlicense}
This Work may be distributed and/or modified under the
conditions of the LaTeX Project Public License 1.3c.
This version of this license is in
```

[2] The absence of indentation at the start of this paragraph is a funny effect due to it immediately following `framed` which itself immediately follows a `verbatim`.

[3] for basic information on this issue, see: http://tex.stackexchange.com/questions/117906/use-of-everyeof-and-endlinechar-with-scantokens

```
> <http://www.latex-project.org/lppl/lppl-1-3c.txt>

and the latest version of this license is in

> <http://www.latex-project.org/lppl.txt>

and version 1.3 or later is part of all distributions of
LaTeX version 2005/12/01 or later.

The Author of this Work is:

- Jean-François Burnol `<jfbu at free dot fr>`

This Work consists of the main source file filecontentsdef.dtx and
the derived files filecontentsdef.sty, filecontentsdef.ins,
filecontentsdef.pdf, filecontentsdef.dvi, README.md.

CHANGE LOG
==========

v1.2 \[2016/09/19\]
-------------------

Initial version.

test: éèàùÉÈÇÀÙÛÎåðñòóôõöœøùúûüýþßŸŽ§
\end{filecontentsdef}
```

The file `filecontentsdef.license` is created with the usual three commentary lines at top of it. And macro `\fcdlicense` contains the verbatim material. It can then be expanded anywhere in the document. Here are some relevant details:

1. the usual special characters are sanitized like in a verbatim environment,
2. spaces become the active character of ascii code `32`, and end of lines are converted into the active character of ascii code `13` (i.e. `^^M`),
3. the tabs `CTRL-I` have been converted to active spaces,
4. the form feeds `CTRL-L` have been converted to blank lines (`^^M^^M`),
5. the characters of ascii code between `128` and `255` have been either given the catcode `letter`, or if they were active (which will be the case with package `inputenc`), they are just inserted in the produced macro as active characters.

Thus what is needed before inserting `\fcdlicense` in the document is to give definitions to the active space and the active `^^M`. LaTeX and TeX both provide `\obeyspaces` and `\obeylines`. For a true verbatim printout, these are usually not enough because spaces at start of lines will disappear, and multiple empty lines give multiple `\par`'s which collapse into a single one (hence no empty line can be observed in the output). The usual `verbatim` environment uses a special definition of `\par` which prevents the disappearance of empty lines, and for the spaces it has

macro \@vobeyspaces which makes the spaces issue \leavevmode so they are not skipped at the start of lines. Let's define:

```
\makeatletter
% this redefines active spaces, but does not make spaces active
\def\niceactivespaces{\@vobeyspaces\catcode32=10\relax}%
\makeatother
\begingroup
% this redefines active end of lines, but does not make them active
  \catcode`\^^M\active %
  \gdef\niceactiveCRs{\def^^M{\leavevmode\par}}%
\endgroup %
```

Then we can issue something like (the output is not shown):

```
{\setlength{\parindent}{1cm}\niceactivespaces\niceactiveCRs\fcdlicense}
```

Notice however this will still allow hyphenation and ligatures, which are usually inhibited in standard verbatim (and also we have not switched to the monospace font.) To emulate exactly what a real verbatim would give, **filecontentsdef** provides \filecontentsprint which is a command with one mandatory argument whose invocation will produce the same as what

\filecontentsprint\macro

```
\begin{verbatim}
<contents of \macro>
\end{verbatim}
```

would have given.[4] The tokens stored in the macro must be of the type described above. As an illustration, here is the output from \filecontentsprint\fcdlicense:

```
This Work may be distributed and/or modified under the
conditions of the LaTeX Project Public License 1.3c.
This version of this license is in

> <http://www.latex-project.org/lppl/lppl-1-3c.txt>

and the latest version of this license is in

> <http://www.latex-project.org/lppl.txt>

and version 1.3 or later is part of all distributions of
LaTeX version 2005/12/01 or later.

The Author of this Work is:

- Jean-François Burnol `<jfbu at free dot fr>`

This Work consists of the main source file filecontentsdef.dtx and
the derived files filecontentsdef.sty, filecontentsdef.ins,
filecontentsdef.pdf, filecontentsdef.dvi, README.md.

CHANGE LOG
==========
```

[4] This is compatible with verbatim.sty's verbatim and hopefully also with other packages modifiying the way the verbatim environment works.

```
v1.2 \[2016/09/19\]
-------------------

Initial version.

test: éèàùÉÈÇÀÙÛÎåðñòóôõöœøùúûüýþßŸŽ§
```

2 Implementation

```
\NeedsTeXFormat{LaTeX2e}[1999/12/01]
\ProvidesPackage{filecontentsdef}
 [2016/09/19 v1.2 filecontents + macro + verbatim (JFB)]
```

Most of the code is still identical to the one in Scott Pakin's filecontents hence to the original one in LaTeX's sources.

```
\begingroup
\catcode`\^^M\active%
\catcode`\^^L\active\let^^L\relax%
\catcode`\^^I\active%
\gdef\filec@ntentsdef#1#2{%
  \let#2\@empty%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck%
    \@latex@warning@no@line%
        {Writing file `\@currdir#1'}%
  \else%
    \@latex@warning@no@line%
        {Overwriting file `\@currdir#1'}%
  \fi%
  \closein\@inputcheck%
  \chardef\reserved@c15 %
  \ch@ck7\reserved@c\write%
  \immediate\openout\reserved@c#1\relax%
  \if@tempswa%
    \immediate\write\reserved@c{%
      \@percentchar\@percentchar\space%
          \expandafter\@gobble\string\LaTeX2e file `#1'^^J%
      \@percentchar\@percentchar\space  generated by the %
        `\@currenvir' \expandafter\@gobblefour\string\newenvironment^^J%
      \@percentchar\@percentchar\space from source `\jobname' on %
         \number\year/\two@digits\month/\two@digits\day.^^J%
      \@percentchar\@percentchar}%
  \fi%
  \let\do\@makeother\dospecials%
```

SP's `filecontents` sets here in the loop all catcodes to `11`, but we need for correct rendering in verbatim that the constructed macro stores active characters as active characters.

We don't check for unusual active characters of ascii code `<128` as this is not done by original or SP's `filecontents`. But if present then they will expand similarly both in the `\write` and in the construction of the macro.

```
  \count@=128\relax%
  \loop%
    \ifnum\catcode\count@=\active%
          \lccode`~\count@%
          \lowercase{\def~{\noexpand~}}%
```

```
    \else%
          \catcode\count@=11 %
    \fi%
    \advance\count@ by \@ne%
    \ifnum\count@<\@cclvi%
  \repeat%
```

The default active ^^L is \outer. But \reserved@b will be def'd with an active ^^L in its replacement text.

```
  \let^^L\relax%
  \edef\E{\@backslashchar end\string{\@currenvir\string}}%
  \edef\reserved@b{\def\noexpand\reserved@b####1\E####2\E####3\relax}%
  \reserved@b{%
    \ifx\relax##3\relax%
      \immediate\write\reserved@c{##1}%
```

This is where the original filecontents is extended to store the parsed material in a macro (in my very first hack I simply patched it to redefine \write to also do the macro storage, but considerations like the one relative to active characters due to inputenc made me decide to re-write the whole thing, hence make a new package.)

Active characters were defined with a single \noexpand in the loop, and this is enough because after each new line is processed the characters it contains are protected from further expansion in the \xdef's. And the single \noexpand is enough also for the \write done above.

The lccode of the tilde is 32 when this gets executed. Multiple form feeds produce the same effect in the macro (insertion of two ^^M per form feed) as in the written out file (via two ^^J).

```
      \toks@\expandafter{#2}%
      {\def^^L{\noexpand^^M\noexpand^^M}\lowercase{\let^^I~}%
       \xdef#2{\the\toks@##1\noexpand^^M}}%
    \else%
      \edef^^M{\noexpand\end{\@currenvir}}%
      \ifx\relax##1\relax%
      \else%
          \@latex@warning{Writing text `##1' before %
             \string\end{\@currenvir}\MessageBreak as last line of #1}%
        \immediate\write\reserved@c{##1}%
```

Same added code as above.

```
        \toks@\expandafter{#2}%
        {\def^^L{\noexpand^^M\noexpand^^M}\lowercase{\let^^I~}%
         \xdef#2{\the\toks@##1\noexpand^^M}}%
      \fi%
      \ifx\relax##2\relax%
      \else%
         \@latex@warning{%
           Ignoring text `##2' after \string\end{\@currenvir}}%
      \fi%
    \fi%
    ^^M}%
  \catcode`\^^L\active%
  \let\L\@undefined%
  \def^^L{\@ifundefined L^^J^^J^^J}%
  \catcode`\^^I\active%
  \let\I\@undefined%
  \def^^I{\@ifundefined I\space\space}%
  \catcode`\^^M\active%
  \edef^^M##1^^M{\noexpand\reserved@b##1\E\E\relax}%
```

We want space characters to be active in the produced macro. We only need to protect them once from expansion.

```
  \catcode32\active\lccode`~32 \lowercase{\def~{\noexpand~}}%
}%
\endgroup
\begingroup
\catcode`\*=11
\gdef\filecontentsdef {\@tempswatrue\filec@ntentsdef}%
\gdef\filecontentsdef*{\@tempswafalse\filec@ntentsdef}%
\global\let\endfilecontentsdef \endfilecontents
\global\let\endfilecontentsdef*\endfilecontents
\gdef\filecontentshere #1{\@tempswatrue
                          \filec@ntentsdef{#1}\filecontentsheremacro}%
\gdef\filecontentshere*#1{\@tempswafalse
                           \filec@ntentsdef{#1}\filecontentsheremacro}%
\gdef\endfilecontentshere{\endfilecontentsdef\aftergroup\filecontents@verbatim}%
\global\let\endfilecontentshere*\endfilecontentshere
```

Package `verbatim.sty` modifies the standard `verbatim` environment. For both the original and the modified version we need to insert an active `^^M` upfront, else an empty first line would not be obeyed. The `verbatim.sty`'s `verbatim` needs that we feed it with the macro expanded once, as it uses active end of lines as delimiters and they thus need to be immediately visible. It also needs an active `^^M` after the `\end{verbatim}`. To avoid to check at `\AtBeginDocument` if package `verbatim.sty` is loaded, we use a slightly tricky common definition. The advantage is that this may help make the code compatible with further packages (I have not looked for them) modifying the `verbatim` environment. For better code readibility I use `^^M%`'s rather than exploiting the active ends of lines here.

```
\catcode`\^^M\active%
\gdef\filecontentsprint #1{\let\filecontentsprint@EOL^^M\let^^M\relax%
   \begingroup\toks@\expandafter{#1}\edef\x{\endgroup%
             \noexpand\begin{verbatim}^^M%
             \the\toks@\@backslashchar end\string{verbatim\string}}\x^^M%
   \filecontentsprint@resetEOL}%
\gdef\filecontentsprint@resetEOL{\let^^M\filecontentsprint@EOL}%
\endgroup
\def\filecontents@verbatim {\filecontentsprint\filecontentsheremacro}%
\def\filecontentsexec #1{\newlinechar13
    \scantokens\expandafter{#1}\newlinechar10\relax}%
\endinput
```