nicefilelist.sty

—

`\listfiles` Alignment for Connoisseurs*

Uwe Lück†

May 20, 2012

Abstract

While longnamefilelist.sty improves LaTeX's `\listfiles` with respect to long base filenames only, nicefilelist.sty can keep separate columns for (i) date (ii) version, and (iii) "caption" (don't write caption text in date column), their alignment not being disturbed by short filename extensions such as `.fd`. This is achieved basing on the monofill package.

Thus nicefilelist is more "powerful" than longnamefilelist, the former however is an "extension" of the latter neither with respect to implementation nor with respect to user interface.

Contents

*This document describes version v0.4 of nicefilelist.sty as of 2012/05/20.

†`http://contact-ednotes.sty.de.vu`

1 Features and Usage

We are describing relations to, ahm, related packages—rather briefly. The latexfileinfo-pkgs package provides a more general overview.

1.1 Relation to longnamefilelist.sty

longnamefilelist.sty equips `\listfiles` with an optional argument for the maximum number of characters in the base filename. By contrast, nicefilelist does not provide arguments for `\listfiles`, rather column widths for basename, extension, and version number are determined by *templates* using monofill.sty. As a "template" for doing this, see the initial settings in Sec. 2.2. (Such settings must precede the `\listfiles command`) So nicefilelist's *user interface* (at present) does not *extend* longnamefilelist's user interface.

Using monofill is a very different approach than the one of longnamefilelist. nicefilelist is more powerful than longnamefilelist, but is not based on it in any way. It does not make sense to load both packages, they just overwrite each other's behaviour of `\listfiles`.

longnamefilelist may become "obsolete" by the present package, unless one finds that its version of `\listfiles` looks fine enough and it is easier to understand and to use than nicefilelist.

1.2 Installing

The file nicefilelist.sty is provided ready, installation only requires putting it somewhere where TEX finds it (which may need updating the filename data base).[1]

1.3 Calling

Below the `\documentclass` line(s) and above `\begin{document}`, you load nicefilelist.sty (as usually) by

```
\usepackage{nicefilelist}
```

Alternatively—e.g., for use with myfilist from the fileinfo bundle, see Sec. 1.4, or in order to include the `.cls` file in the list—you may load it by

```
\RequirePackage{nicefilelist}
```

before `\documentclass` or when you don't use `\documentclass`.

[1] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

As of v0.4, there is a package option `[r]` in order to place strings like '`r0.4`' in the column reserved for version numbers. You get this functionality by

```
\usepackage[r]{nicefilelist}
```

or

```
\RequirePackage[r]{nicefilelist}
```

See Section 2.5 for more information.

1.4 Usage and Sample with myfilist.sty

In order to get a reduced and/or rearranged list of used files with the myfilist package, `nicefilelist.sty` must be loaded earlier than `myfilist.sty`. This is due to a kind of limitation of the latter, it *issues* `\listfiles` (TODO). Therefore `\listfiles` must be modified earlier—or *issued* earlier, in this case the `\listfiles` in `myfilist.sty` does nothing. The file `SrcFILEs.txt` accompanying the distribution of longnamefilelist, e.g., can be generated by running the following file `srcfiles.tex` with LaTeX:

```
\ProvidesFile{srcfiles.tex}[2012/03/23
                            file infos -> SrcFILEs.txt]
\RequirePackage{nicefilelist}
%% INSERT MODIFICATIONS OF INITIAL
%% 'nicefilelist'/'monofill' SETTINGS HERE!
\RequirePackage{myfilist}
%% documentation:
\ReadFileInfos{nicefilelist}
%% demonstration:
\ReadFileInfos{provonly.fd,wrong.prv,empty.f}
% \ReadFileInfos{utopia.xyz}
%% present file:
\ReadFileInfos{nicefilelist}
\ReadFileInfos{srcfiles}
\ListInfos[SrcFILEs.txt]
```

Note the lines where to place **custom** modifications of settings for alignment (Sec. 2.2) or failure displays (Sec. 2.3).

The previous code mentions the following files:

`provonly.fd` has a proper `\ProvidesFile` line without date, for seeing what happens in the date and version columns. It also was a test for the case that there are fewer characters than a date has, and there is no blank space.

`wrong.prv` has a `\ProvidesFile` line with wrong file name.

`empty.f` just is an empty file.

`utopia.xyz` is not present at all, you get an error when you remove the comment mark.

Moreover, my `.tex` files have dates, but not version numbers, so you see what happens then:

```
      *File List*
nicefilelist.sty  2012/03/23  v0.1   more file list alignment (UL)
    monofill.sty  2012/03/19  v0.1a  monospace alignment (UL)
    myfilist.sty  2011/01/30  v0.3a  \listfiles -- mine only (UL)
    readprov.sty  2010/11/27  v0.3   file infos without loading (UL)
nicefilelist.tex  2012/03/23   --    documenting nicefilelist.sty
    provonly.fd    --  -- --   --    such
       wrong.prv   * NOT FOUND *
       empty.f     * NOT FOUND *
    srcfiles.tex  2012/03/23   --    file infos -> SrcFILEs.txt
     ***********

 List made at 2012/03/23, 10:31
 from script file srcfiles.tex
```

2 Implementation

2.1 Package File Header (Legalese)

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{nicefilelist}[2012/05/20 v0.4
                               more file list alignment (UL)]

%% Copyright (C) 2012 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% We did our best to help you, but there is NO WARRANTY.
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

2.2 Alignment Settings

We use the monofill package for alignment of plain text:

```
\RequirePackage{monofill}
```

See its documentation for details.

We support three alignment "fields" according to the terminology of monofill. Their ids are `f-base` for base filenames, `f-ext` for filename extensions, and `f-version` for the revision version id of a file as read from `\ProvidesFile`, `\ProvidesPackage`, or `\ProvidesClass` command in the file. Initial settings for them are following. For modifying them, load nicefilelist.sty, then type your own settings, then issue `\listfiles` or load myfilist.sty.

```
\MFfieldtemplate{f-base}{nicefilelist}
\MFfieldtemplate{f-ext}{tex}
\MFfieldtemplate{f-version}{v0.11a}
```

We are not supporting version numbers greater than 9 at present—sorry! (TODO)

`\NFLspaceI`, `\NFLspaceII`, and `\NFLspaceIII` determine the space between the four columns for names, dates, versions, and "captions":

```
\newcommand*{\NFLspaceI}  { \space}
\newcommand*{\NFLspaceII} { \space}
\newcommand*{\NFLspaceIII}{ }
```

2.3 Failure Displays

`\NFLnodate` is displayed in place of a file date if it seems not to be given (configurable):

```
\newcommand*{\NFLnodate}{ -- \space-- --}
```

`\NFLnoversion` likewise—however, for alignment, each wanted space must be specified as `\space` (not just a code blank space). It may need adjustment (by `\renewcommand`) when `\MFfieldtemplate{f-version}` is modified:

```
\newcommand*{\NFLnoversion}{\space--}
```

`\NFLnotfound` is for files with wrong or no `\Provides...` command:

```
\newcommand*{\NFLnotfound}{ * NOT FOUND *}
```

2.4 Safe Tests

For fairly safe tests, we briefly use an exotic version of `Q` (similarly to ifmptarg):

```
\catcode`\Q=7 \let\NFL@criterion=Q \catcode`\Q=11
```

It appears to me that expandable tests like the ones employed here never are perfectly safe; you only can say that it is safe with a source meeting certain conditions. fifinddo originally was made for "plain text," to be read from files without assigning TEX's special category codes. *Here* we assume that the source (text in `\Provides...` arguments) will never contain such a "funny `Q`".

2.5 Package Option [r]

v0.4 offers package option [r] that allows strings with r in place of v, for "release." \NFL@v@digit's definition therefore depends ... we use \@listfiles for a "message" there. For the original restricted functionality, it expands to \NFL@false.

```
\def\@listfiles{\noexpand\NFL@false}
```

Package option [r] carries out another test instead. See the accompanying file SrcFILEs.txt to see the effect. TODO: update example!?

```
\DeclareOption{r}{%
  \def\@listfiles{%
    {\noexpand\NFL@ifx@kbl##1r%
        {\noexpand\NFL@digits##2\noexpand\@nnil}%
        \noexpand\NFL@false}%
  }%
}
\ProcessOptions
```

2.6 Redefining \listfiles

Similarly to original LaTeX, \listfiles carries almost everything that is needed for the file list only:

```
\renewcommand*{\listfiles}{%
  \let\listfiles\relax
```

—this clears memory. Now LaTeX doesn't collect file names for \listfiles when \@listfiles is undefined, therefore

```
%   \let\@listfiles\relax
```

... postponed for v0.4 ...

\@dofilelist is executed by the standard LaTeX \enddocument macro or by \ListInfos from the myfilist package.

```
    \def\@dofilelist{%
```

"Title:"

```
      \typeout{^^J           %% trick 2012/03/29 vv
               \MFrightinfield{*File Lis}{f-base}t*}%
      \@for\@currname:=\@filelist\do{%
```

This starts the loop through the list of files

```
        \filename@parse\@currname
        \edef\filename@ext{%
           \ifx\filename@ext\relax tex\else\filename@ext\fi}%
```

Like LaTeX's \reserved@b:

```
50          \expandafter\let\expandafter\@tempb
51               \csname ver@\filename@base.\filename@ext\endcsname
```

According to `source2e.pdf`, `\filename@area` may be a directory. Trying support of this is seems to be a new feature with v0.2—not tested, TODO!

```
52          \edef\@tempa{\filename@area\filename@base}%
```

Actually I would like to be able to do even the filename parsing expandably—for all systems, `texsys.cfg`!?? TODO

```
53          \typeout{%
```

Now all parsing and checking must be expandable.

```
54            \NFL@make@macro@arg\MFrightinfield\@tempa      {f-base}.%
55            \NFL@make@macro@arg\MFleftinfield \filename@ext{f-ext}%
56            \NFLspaceI
57            \NFL@ifx@kbl\@tempb\relax\NFLnotfound{%
58              \NFL@make@macro@arg\NFL@space@split\@tempb
59                                                 \NFL@maybe@three
60                                                 \NFL@date@or@rest
61            }%
62          }%
63        }%
```

The line of stars:

```
64        \typeout{                          %% trick vvv 2012/03/29
65                 \MFrightinfield{********}{f-base}***^^J}%
```

TODO or more stars as with longnamefilelist?

```
66      }%
```

This finishes the definition of `\@dofilelist`.

`\NFL@make@macro@arg`⟨*cmd-1*⟩⟨*cmd-2*⟩

results in ⟨*cmd-1*⟩`{`⟨*t-list*⟩`}` where ⟨*t-list*⟩ is the one-step expansion of ⟨*cmd-2*⟩:

```
67      \def\NFL@make@macro@arg##1##2{\expandafter##1\expandafter{##2}}%
```

`\NFL@space@split{`⟨*token-list*⟩`}{`⟨*spaced*⟩`}{`⟨*unspaced*⟩`}` passes prefix and suffix as arguments to ⟨*spaced*⟩ if a space token is within ⟨*token-list*⟩, otherwise ⟨*unspaced*⟩ gets the original ⟨*token-list*⟩ as single argument. The latter is useful here where ⟨*token-list*⟩ becomes visible only by an `\expandafter`. The following construction is discussed more generally in the bitelist package.

```
68      \def\NFL@space@split##1{%
69        \NFL@return@space@split##1\@nil: \NFL@criterion\@nil\@nil@{##1}}%
```

`\NFL@return@spaces@split` essentially has *three* parameters delimited by ␣, `\@nil`, and `\@nil` again.

```
70    \def\NFL@return@space@split##1 ##2\@nil##3\@nil@###4###5##6{%
71      \NFL@ifx@kbl\NFL@criterion{##2}%
```

If `#2` is empty, `\NFL@ifx@kbl` (as of v0.3) compares `\NFL@criterion` (criterion indicating "unspaced") with `\expandafter`. This only happens when the space is the last thing in ⟨*token-list*⟩, and ⟨*spaced*⟩ is chosen correctly.

```
72      {##6{##4}}{##5{##1}{##2}}}%
```

`\NFL@ifx@kbl{`⟨*token*⟩`}{`⟨*maybe-token*⟩`}{`⟨*ifx*⟩`}{`⟨*unlessx*⟩`}` as of v0.3 should save some tokens, in some longer run, especially if we want to add nestings—cf. `source2e.pdf` for "Kabelschacht."

```
73    \def\NFL@ifx@kbl##1##2{%
74      \ifx##1##2\expandafter  \@firstoftwo
75        \else   \expandafter \@secondoftwo \fi}%
```

Dealing with `\NFL@date@or@rest{`⟨*token-list*⟩`}` before `\NFL@maybe@three`:

```
76    \def\NFL@date@or@rest##1{%
77      \NFL@if@date{##1}{##1}{\NFL@no@date@version##1}}%
```

`\NFL@if@date{`⟨*token-list*⟩`}{`⟨*yes*⟩`}{`⟨*no*⟩`}` ...

```
78    \def\NFL@if@date##1{\NFL@slashes##1\NFL@xi xyzxyzxyzx\@nil}%
```

`\NFL@slashes` checks that there are slashes at the expected places:

```
79    \def\NFL@slashes##1##2##3##4##5##6##7##8{%
80      \NFL@ifx@kbl##5/%
81        {\NFL@ifx@kbl##8/\NFL@ten@only\NFL@false}%
82        \NFL@false
```

This especially happens when ⟨*token-list*⟩ is empty. Digit candidates back:

```
83      {##1##2##3##4##6##7}}%
```

If the word is a date, we now have taken 6 of the 8 digits.

`\NFL@ten@only{`⟨*digits*⟩`}`⟨*digit*⟩⟨*digit*⟩`Q`

takes the two remaining and then a thing that should be `Q` in the funny sense of Sec. 2.4.

```
84    \def\NFL@ten@only##1##2##3##4{%
85      \NFL@ifx@kbl\NFL@xi##4\NFL@digits\NFL@false
```

Finally checking digits:

```
86      ##1##2##3\@nnil}%
```

`\NFL@digits`⟨*token*⟩ is a loop through single tokens:

```
87    \def\NFL@digits##1{%
88      \NFL@ifx@kbl##1\@nnil\NFL@true{%
89        \NFL@if@digit@code##1<0\NFL@false{%
90          \NFL@if@digit@code##1>9\NFL@false\NFL@digits
91        }%
92      }%
93    }%
```

`\NFL@if@digit@code`⟨*char-1*⟩⟨*relation*⟩⟨*char-2*⟩⟨*fits*⟩⟨*bad*⟩:

```
94    \def\NFL@if@digit@code##1##2##3{%
95      \ifnum`##1##2`##3 \expandafter  \@firstoftwo
96        \else           \expandafter \@secondoftwo \fi}%
```

`\NFL@false` skips further candidates and dummies and chooses ⟨*no*⟩:

```
97    \def\NFL@false##1\@nil{\@secondoftwo}%
```

`\NFL@true` skips further candidates and dummies and chooses ⟨*yes*⟩:

```
98    \def\NFL@true##1\@nil{\@firstoftwo}%
```

We don't support version without date, therefore run `\NFL@no@date@version` as soon as we find that the file info does not start with a date:

```
99    \def\NFL@no@date@version{%
100     \NFLnodate\NFLspaceII\NFLnoversion@\NFLspaceIII}%
```

`\NFLnoversion@` adds filler to `\NFLnoversion`:

```
101   \def\NFLnoversion@{%
102     \NFL@make@macro@arg\NFL@place@version\NFLnoversion}%
```

`\NFL@maybe@three{`⟨*word-1*⟩`}{`⟨*rest*⟩`}` looks whether ⟨*word-1*⟩ is a date. If it is, it is written to screen, and then we look if ⟨*rest*⟩ contains a version id. Otherwise "⟨*word-1*⟩␣⟨*rest*⟩" is considered a "caption" only.

```
103   \def\NFL@maybe@three##1##2{%
104     \NFL@if@date{##1}%
105                 {##1\NFLspaceII
106                  \NFL@space@split{##2}%
107                                  \NFL@maybe@version@rest
108                                  \NFL@version@or@rest}%
109                 {\NFL@no@date@version##1 ##2}}%
```

`\NFL@version@or@rest{`⟨*token-list*⟩`}`:

```
110   \def\NFL@version@or@rest##1{%
111     \NFL@if@version{##1}%
112                    {\NFL@place@version{##1}}%
113                    {\NFLnoversion@\NFLspaceIII##1}}%
```

`\NFL@if@version{`⟨*token-list*⟩`}{`⟨*yes*⟩`}{`⟨*no*⟩`}`:

```
    \def\NFL@if@version##1{\NFL@v@digit##1xy\@nil}%
```

TODO: At applications you see how some tokens could be saved. On the other hand, the macros are more transparent in the present way.

`\NFL@v@digit{⟨t1⟩}{⟨t2⟩}{⟨rest⟩}` checks whether the first thing is a `v` and the second a digit—unless package option `[r]` was chosen. v0.4 uses `\edef` for choosing:

```
    \edef\NFL@v@digit##1##2##3\@nil{%
      \noexpand\NFL@ifx@kbl##1v%
          {\noexpand\NFL@digits##2\noexpand\@nnil}%
```

`\@listfiles` will either expand to the original `\NFL@false` or to a test on `r`:

```
          \@listfiles
      \noexpand\@nil}%
    \let\@listfiles\relax
```

`\NFL@place@version{⟨token-list⟩}` adds filler to version id:

```
    \def\NFL@place@version##1{\MFleftinfield{##1}{f-version}}%
```

`\NFL@maybe@version@rest{⟨list-1⟩}{⟨list-2⟩}`:

```
    \def\NFL@maybe@version@rest##1##2{%
      \NFL@if@version{##1}%
                     {\NFL@place@version{##1}\NFLspaceIII##2}%
                     {\NFLnoversion@\NFLspaceIII##1 ##2}}%
}
```

2.7 Leaving the Package File

```
\endinput
```

2.8 VERSION HISTORY

```
v0.1   2012/03/20   started
       2012/03/22   almost ready
       2012/03/23   debugging; \NFLspaceI etc.;
                    documentation completed

v0.2   2012/03/24   file info processed by \typeout - start
       2012/03/25   trying, debugging
       2012/03/26   continued; \NFL@place@version, \NFLnoversion@;
                    works, reordered; another fix about Q -> \@empty
       2012/03/27   undone the latter, explained; improved remarks on
                    \@listfiles
       2012/03/29   alignment of title/stars with base<11

v0.30  2012/05/18f. \NFL@ifx@kbl in \NFL@return@space@split
       2012/05/20   all \ifx reimplemented, old code kept
```

```
143         STORED INTERNALLY
144   v0.31 2012/05/20   removing old code - STORED INTERNALLY
145   v0.32 2012/05/20   removing \NFL@xpxpxp; replacing \NFL@after@false
146                      by \NFL@ifnum@kbl, keeping old code
147         STORED INTERNALLY
148   v0.33 2012/05/20   removing old code; added 3 %s
149         STORED INTERNALLY
150   v0.4  2012/05/20   option [r]
151
```

3 Credits

1. It was Martin Muench who pointed out the shortcomings of longnamefilelist that the present package addresses—thanks!

2. For Alois Kabelschacht—whose idea in TUGboat **8** #2[2] is used for v0.3—cf. the dowith documentation.

4 Missing

1. The package once might provide keyval-style optional arguments for `\listfiles` or even call `\listfiles` automatically with keyval package options.

2. Another idea from Martin Muench: wrapping inside caption column. Can hardwrap help?

[2] "`\expandafter` vs. `\let` and `\def` in Conditionals and a Generalization of PLAIN's `\loop`," TUGboat Vol. 8 (1987), No. 2, pp. 184f. (`tug.org/TUGboat/tb08-2/tb18kabel.pdf`)