nicefilelist.sty

—

`\listfiles` Alignment for Connoisseurs*

Uwe Lück

February 13, 2023

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.

v0.7 offers a package option `[wrap]` for automatic word wrapping within the caption column (using the hardwrap package), so filenames and captions can be quite long without disturbing alignment.

As opposed to the dateiliste package, this is about the *plain text* output in the `.log` file or, with myfilist, as a stand-alone plain text file.

Related packages:: Cf. latexfileinfo-pkgs.

Keywords: Package management, document management, plain text output

Contents

*This document describes version v0.9b of nicefilelist.sty as of 2023/02/13.

1 Features and Usage

Additionally or also "complementarily" to the presentation given here, the functionality of the package is summarized in the file `latexfileinfo_pkgs.htm` from the latexfileinfo-pkgs, in a comparison with packages resembling nicefilelist in certain respects.

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 section 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).

1.3 Calling

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

```
\usepackage{nicefilelist}
```

or by

```
\usepackage[⟨options⟩]{nicefilelist}
```

where ⟨*options*⟩ may be `r`, `wrap`, and/or `autolength`—see summaries in sections 1.4 and 2.4 on the package options and an example in section 1.5.2. Alternatively—e.g., for use with myfilist from the fileinfo bundle (in a "TEX script"), see section 1.5, or in order to include the `.cls` file in the list—you may load it by

```
\RequirePackage{nicefilelist}
```

or by

```
\RequirePackage[⟨options⟩]{nicefilelist}
```

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

1.4 Choosing Settings

1.4.1 The Columns, Their Widths, and Their "Missing" Content

The nicefilelist package considers the listing from '`\listfiles`' a five-column table, the columns being (reserved for)
(i) the base filename,
(ii) the filename extension,
(iii) the date,
(iv) the version (or with option '`[r]`': the release) number, and
(v) the caption
of a LATEX source file. The filename base column is right-adjusted, the other ones are left-adjusted. Date, version, and caption are made up from the ⟨*f-info*⟩ argument in

`\Provides`⟨*f-type*⟩`{`⟨*f-base*⟩`.`⟨*f-ext*⟩`}[`⟨*f-info*⟩`]`

where ⟨*f-type*⟩ is '`Class`', '`Package`', or '`File`'.

The fixed usual format '`YYYY/MM/DD`' or '`YYYY-MM-DD`' (accepted by LATEX since 2017-03-08) for the date is assumed; in fact, when ⟨*f-info*⟩ doesn't start according to this format, it is assumed that no date is given, and some "missing" text will appear in the "date" column, determined by a macro `\NFLnodate`.

Using the filemod package, it is possible to use the date of a file which does not explicitly declare a date:

```
\makeatletter
\RequirePackage{filemod}%
\renewcommand*\thefilemoddate[3]{#1-#2-#3}
\renewcommand*{\NFLnodate}{\filemodprintdate{%
                \filename@area\filename@base.\filename@ext}}
\makeatother
\renewcommand*{\NFLnotfound}{%
                \NFLnodate\NFLspaceII\NFLnoversion\NFLspaceIII}
```

The version number (or "string") must follow in format '`v`⟨*digit*⟩`.`⟨*digits*⟩', otherwise some "missing" text appears in the "version" column, determined by a macro `\NFLnoversion`. What remains is placed in the "caption" column. `\NFLnotfound` determines an alternative filling in the case that ⟨*f-info*⟩ cannot be obtained. See the default settings for these "failure" texts in section 2.3.

The column widths for filename base and extension and the column width for version or release are determined using the monofill package. They have "field identifiers" `f-base`, `f-ext`, and `f-version` respectively. The respective widths are determined by templates ⟨*longest*⟩ in

`\MDfieldtemplate{`⟨*field-id*⟩`}{`⟨*longest*⟩`}`

See section 2.2 for the default settings. Probably only adjusting the width for *base* filenames is required in real life, see the example in section 1.5.2. (But there exist extensions .info and .mkii and versions v4.6.3.1 (fontawesome.sty).) Option '`autolength`' measures the respective length of the field and writes it into the .aux file for use in the next compilation run. This option requires LaTeX format-version at least 2022-11-01.

The spaces between the columns are determined by macros `\NFLspaceI`, `\NFLspaceII`, and `\NFLspaceIII`, see section 2.2 for the defaults.

1.4.2 The Caption Column

The width of the caption column (unfortunately) is determined by the stuff enumerated above and the width of the console output window or screen. With long filenames and long captions, the result may look poor. the *characters* that don't fit into the line may continue at left end of the window or screen, disturbing the appearance of a "table"—unless you use package option `[wrap]`. The latter requires the hardwrap package by Will Robertson and Kevin Godby ("not invented here"). This package tries to determine the screen width by some subtle tests, and until it finds something better, it assumes a width of 80 characters (I suppose). hardwrap does *word wrapping*, i.e., it doesn't just put *characters* not fitting into the next line, but entire *words*. Moreover, it allows inserting some "newline sequence" before the first word that is too much, and we use this feature here to put the next word into the *caption column* rather than at the beginning of the next line. (Details and implementation are in section 2.4.)

If you are not happy with the column width that hardwrap chooses, but want to assume your own width ⟨*max-line-chars*⟩ (e.g., your width, measured by your doctor, divided by the width of one character), compute its difference ⟨*max-line-chars-minus-one*⟩ to 1 (maybe by your electronic calculator, or an emulation, or a Lua script, cf. lualatex-doc, or by bigintcalc), and enter the hardwrap instruction

`\setmaxprintline{`⟨*max-line-chars-minus-one*⟩`}`

when hardwrap or nicefilelist have been loaded *and* before the internal macro `\@dofilelist` is run (which happens at the end of the document or when myfilist's `\ListInfos` is issued, for instance).

1.5 Usage and Samples with myfilist.sty

1.5.1 Basically

In order to get a reduced and/or rearranged list of file infos 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—first—distribution of nicefilelist was 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 (section 2.2) or failure displays (section 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*
-----RELEASE.---   --  -- --   --     --
nicefilelist.RLS  2023/02/13  v0.9b  bug-fix
-----PACKAGE.---   --  -- --   --     --
nicefilelist.sty  2023/02/13  v0.9b  more file list alignment (UL)
------DOCSRC.---   --  -- --   --     --
nicefilelist.tex  2023/02/13   --    documenting nicefilelist.sty
    srcfiles.tex  2023/02/13   --    file infos -> SrcFILEs.txt
--------DEMO.---   --  -- --   --     --
    provonly.fd    --  -- --   --    no date, no version, but a lot of info,
                                     look how that is wrapped!
       wrong.prv   * NOT FOUND *
       empty.f     * NOT FOUND *
--------USED.---   --  -- --   --     --
    hardwrap.sty  2011/02/12  v0.2   Hard wrap messages
    myfilist.sty  2012/11/22  v0.71  \listfiles -- mine only (UL)
    readprov.sty  2012/11/22  v0.5   file infos without loading (UL)
    fifinddo.sty  2012/11/17  v0.61  filtering TeX(t) files by TeX (UL)
     makedoc.sty  2012/08/28  v0.52  TeX input from *.sty (UL)
    niceverb.sty  2015/11/21  v0.62  minimize doc markup (UL)
    texlinks.sty  2015/07/20  v0.83  TeX-related links (UL)
     makedoc.cfg  2013/03/25   --    documentation settings
    mdoccorr.cfg  2012/11/13   --    'makedoc' local typographical corrections
-not-so-much.---   --  -- --   --     --
   kvsetkeys.sty  2022-10-05  v1.19  Key value parser (HO)
     ***********

 List made at 2023/02/13, 19:17
 from script file srcfiles.tex
```

1.5.2 More Generally and Shorthand

In the above example, the myfilist command '`\EmptyFileList`' was missing—it was not intended there. Usually however, it *is* intended, i.e., the following sequence of lines is wanted:

```
\RequirePackage[r]{nicefilefilelist}
\MFfieldtemplate{f-base}[⟨longest-name⟩]
\RequirePackage{myfilist}
\EmptyFileList[⟨read-again-files⟩]
```

Here you also see usage of package option [r] for release numbers and the adjustment

\MFfieldtemplate{f-base}{⟨*longest-name*⟩}

according to section 2.2.

With v0.5, the last three code lines in the snippet above can be replaced by

\MaxBaseEmptyList{⟨*longest-name*⟩}[⟨*read-again-files*⟩]

—"optionally" without '[⟨*read-again-files*⟩]'. This may save the user from worrying about usage with myfilist.

nicefilelist formats file lists nicely even when base filenames have eight characters at most, what LaTeX's original \listfiles was made for. v0.6 simplifies this case by a star version of \MaxBaseEmptyList:

\MaxBaseEmptyList*

works like \MaxBaseEmptyList{nicefile} (eight characters)—still, optional [⟨*read-again-files*⟩] may follow. This feature is demonstrated with inputtrc v/r0.3.

1.5.3 Sample with Wrapped Caption Column

The most recent version of the accompanying 'SrcFILEs.txt' contains the following:

```
     *File List*
-----RELEASE.---   --  -- --   --     --
nicefilelist.RLS  2023/02/13  v0.9b  bug-fix
-----PACKAGE.---   --  -- --   --     --
nicefilelist.sty  2023/02/13  v0.9b  more file list alignment (UL)
------DOCSRC.---   --  -- --   --     --
nicefilelist.tex  2023/02/13   --    documenting nicefilelist.sty
    srcfiles.tex  2023/02/13   --    file infos -> SrcFILEs.txt
--------DEMO.---   --  -- --   --     --
    provonly.fd    --  -- --   --    no date, no version, but a lot of info,
                                     look how that is wrapped!
       wrong.prv   * NOT FOUND *
       empty.f     * NOT FOUND *
--------USED.---   --  -- --   --     --
    hardwrap.sty  2011/02/12  v0.2   Hard wrap messages
    myfilist.sty  2012/11/22  v0.71  \listfiles -- mine only (UL)
    readprov.sty  2012/11/22  v0.5   file infos without loading (UL)
    fifinddo.sty  2012/11/17  v0.61  filtering TeX(t) files by TeX (UL)
     makedoc.sty  2012/08/28  v0.52  TeX input from *.sty (UL)
    niceverb.sty  2015/11/21  v0.62  minimize doc markup (UL)
    texlinks.sty  2015/07/20  v0.83  TeX-related links (UL)
     makedoc.cfg  2013/03/25   --    documentation settings
    mdoccorr.cfg  2012/11/13   --    'makedoc' local typographical corrections
-not-so-much.---   --  -- --   --     --
   kvsetkeys.sty  2022-10-05  v1.19  Key value parser (HO)
     **********

 List made at 2023/02/13, 19:17
 from script file srcfiles.tex
```

This exemplifies

1. **wrapping** of 'provonly.fd''s and kvsetkeys.sty file info within the **caption** column using nicefilelist's '[wrap]' option,
2. inserted "**comments**" from myfilist's '\FileListRemark',
3. a file 'nicefilelist.RLS' for a **release summary**. This is to track what has happened most recently, whether the most recent release has been installed (system-wide), or (for me) whether most recent versions of package and documentation have been released. When such an '.RLS' file is installed together with packages in the 'tex' subtree of a TDS, the release summary can be accessed quickly as a **terminal display** by one of the packages ltxfileinfo, latexfileversion, or typeoutfileinfo. One aim of the '[wrap]' option is allowing longer "release captions" (looking fine in the package file list) than fit into a small part of a single line.

The above 'SrcFILEs.txt' has been generated from the following version of the TEX script 'srcfiles.tex':

```
\ProvidesFile{srcfiles.tex}
             [2023/02/13 file infos -> SrcFILEs.txt]
\RequirePackage[r,wrap]{nicefilelist}
\RequirePackage{filedate}
\MaxBaseEmptyList{nicefilelist}
                 [nicefilelist.sty,%
                  readprov.sty,myfilist.sty,%
                  hardwrap.sty]
\FileListRemark[ -- ]{-----RELEASE.---}
\ReadFileInfos{nicefilelist.RLS}
\FileListRemark[ -- ]{-----PACKAGE.---}
\ReadPackageInfos{nicefilelist}
\FileListRemark[ -- ]{------DOCSRC.---}
\ReadFileInfos{nicefilelist,srcfiles.tex}
\FileListRemark[ -- ]{--------DEMO.---}
\ReadFileInfos{provonly.fd,wrong.prv,empty.f}
%\ReadFileInfos{utopia.xxx}
%\FileListRemark[ -- ]{DOCUTILITIES.---}
%\FileListRemark[ -- ]{usedNICETEXT.---}
\FileListRemark[ -- ]{--------USED.---}
\ReadPackageInfos{hardwrap,
                  myfilist,readprov,
                  fifinddo,makedoc,niceverb,texlinks}
\ReadFileInfos{makedoc.cfg,mdoccorr.cfg}
\FileListRemark[ -- ]{-not-so-much.---}
```

```
\ReadPackageInfos{kvsetkeys}
%\NoStopListInfos[SrcFILEs.txt]
%\EqualityMessages
\CheckDateOfPDFmod{nicefilelist.sty}
\CheckDateOfPDFmod{nicefilelist.tex}
\CheckDateOfPDFmod{nicefilelist.RLS}
\CheckDateOfPDFmod{srcfiles.tex}
%\stop
\NoBottomLines \ListInfos[SrcFILEs.txt]
```

---

2 Implementation

2.1 Package File Header (Legalese)

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]% Option autolength needs 2022-11-01!
\ProvidesPackage{nicefilelist}[2023/02/13 v0.9b
                               more file list alignment (UL)]
%% Copyright (C) 2012 Uwe Lück (deceased June 2020)
%%
%% This work may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either
%% version 1.3c of this license or (at your option) any later
%% version. This version of this license is in
%%    https://www.latex-project.org/lppl/lppl-1-3c.txt
%% and the latest version of this license is in
%%    https://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of
%% LaTeX version 2005-12-01 or later.
%%
%%
%%
%%
```

2.2 Alignment Settings

We use the monofill package for alignment of plain text:

```
\RequirePackage{monofill}[2012/10/29]
```

See its documentation for details. The `[wrap]` option provided by nicefilelist v0.7 requires monofill v0.2 as of 2012-10-29.

We support three alignment "fields" according to the terminology of monofill. Their ids are `f-base` for base file-names, `f-ext` for file-name 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}
```

`\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 Package Options

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. TO DO: update example!?

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

v0.7 offers package option `[wrap]` for automatic wrapping within the "captions" column, based on Will Robertson's and Kevin Godby's hardwrap package. The difference between this option and the functionality without is controlled by the macro `\NFL@filerow`. *Without* it expands to `\typeout`

```
\newcommand*{\NFL@filerow}{\typeout}
```

—`\let` doesn't work with myfilist's redefinition of `\typeout`. *With* `[wrap]`, `\NFL@filerow` applies hardwrap's `\HardWrap`:

```
\newif\ifNFLwrap%
\DeclareOption{wrap}{%
  \NFLwraptrue%
  \renewcommand*{\NFL@filerow}[1]{%
    \HardWrap\typeout\hw@maxprintline\relax{^^J%
      \MFrightinfield\space{f-base} %
      \MFleftinfield \space{f-ext}%
      \NFLspaceI\@spaces\space\@spaces\space \NFLspaceII
      \MFrightinfield\space{f-version}\NFLspaceIII}{%
      #1}}%
```

Alignment of file-names with hardwrap seems to need

```
    \renewcommand*{\MFfillelement}{\MFotherspace}
```

from monofill v0.2.

```
}
```

The display width is controlled by hardwrap's counter `|\hw@maxprintline|`. Unless hardwrap finds something special, its content is 79, corresponding to a display width of 80 characters (I believe—counting the leftmost character as '0', as editors like to do). You can choose a different content value ⟨*max-char-col*⟩ by hardwrap's

`\setmaxprintline{`⟨*max-char-col*⟩`}`

v0.9a offers package option `[autolength]` for automatic setting of lengths of base file-name, extension, and version.

```
\newif\ifNFLautolength

\DeclareOption{autolength}{% v0.9a 2023-01-08
```

We just measure the respective length and write it into the .aux file for use in the next compilation run. The xstring package requires e-TEX. `\AddToHook{enddocument/afterlastpage}` ensures that the nicefilelist package still works when `\AtEndDocument` would be gone. The `\ver@@`-fix for the file-list was introduced 2022. Thus instead of LaTeX format version 1994/12/01 option [autolength] requires format-version 2022-11-01 or newer. (2018 might be sufficient, but testing was only done with a current format.)

```
    \@ifl@t@r\fmtversion{2022/11/01}{
      % would have understood
      % \IfFormatAtLeastTF{2022-11-01}{<true code>}{<false code>}
      \NFLautolengthtrue
      }{\PackageError{nicefilelist}{%
```

```
58          Option autolength needs newer LaTeX format%
59        }{Needed LaTeX format version: 2022-11-01 or newer.\MessageBreak%
60          Found  LaTeX format version: \fmtversion.\MessageBreak%
61          To use option autolength update your TeX distribution.%
62        }
63      }
64    }
65
66  \ProcessOptions
67
```

The next `\if` is to check whether `[wrap]` has been demanded and hardwrap is needed:

```
68  \ifNFLwrap\RequirePackage{hardwrap}\fi
69
70  \ifNFLautolength
71    \RequirePackage{xstring}[2023-01-14]% v1.85 String manipulations (CT); needs e-TeX
72    \AddToHook{enddocument/afterlastpage}{%
73      \xdef\NFLbaselengthmax{nicefilelist}%
74      \xdef\NFLextlengthmax{sty}%
75      \xdef\NFLversionlengthmax{v0.9b}%
76      \xdef\NFLcaptionlengthmax{more file list alignment (UL)}%
77      \xdef\NFLbaselengthtmp{0}%
78      \xdef\NFLextlengthtmp{0}%
79      \xdef\NFLversionlengthtmp{0}%
80      \xdef\NFLcaptionlengthtmp{0}%
81      \xdef\NFLtotallengthtmp{0}%
82      \@for\@currname:=\@filelist\do{%
83        % This starts the loop through the list of files.
84        \filename@parse\@currname%
85        \edef\filename@ext{\ifx\filename@ext\relax tex\else\filename@ext\fi}%
86        \StrLen{\filename@base}[\NFLbaselengthcurrent]%
87        \ifnum \NFLbaselengthcurrent > \NFLbaselengthtmp \relax%
88          {\xdef\NFLbaselengthtmp{\NFLbaselengthcurrent}%
89           \xdef\NFLbaselengthmax{\filename@base}%
90          }%
91        \fi%
92        \StrLen{\filename@ext}[\NFLextlengthcurrent]%
93        \ifnum \NFLextlengthcurrent > \NFLextlengthtmp \relax%
94          {\xdef\NFLextlengthtmp{\NFLextlengthcurrent}%
95           \xdef\NFLextlengthmax{\filename@ext}%
96          }%
97        \fi%
98        \expandafter\let\expandafter\@NFLtempb\csname ver@\filename@base.\filename@ext\endcsname%
99        \StrBetween[1,2]{\@NFLtempb}{ }{ }[\@NFLtempc]%
100       % assuming date space version space description
101       \IfBeginWith{\@NFLtempc}{v}{%
102         % fails if version omitted and description starts with v - evil!
103         \xdef\filename@version{\@NFLtempc}}{%
```

```
104           % else: empty or unusual format
105           % Packages that \relax their \ver@... (e.g., fontenc)
106           % can use \ver@@... to store the version information instead:
107           \expandafter\let\expandafter\@NFLtempd\csname ver@@\filename@base.\filename@ext\endcsname%
108           \StrBetween[1,2]{\@NFLtempd}{ }{ }[\@NFLtempc]%
109           % assuming date space version space description again
110           \IfBeginWith{\@NFLtempc}{v}{\xdef\filename@version{\@NFLtempc}%
111            }{% else: no usual version found
112              \xdef\filename@version{}%
113             }%
114           }%
115         \StrLen{\filename@version}[\NFLversionlengthcurrent]%
116         \ifnum \NFLversionlengthcurrent > \NFLversionlengthtmp \relax%
117           {\xdef\NFLversionlengthtmp{\NFLversionlengthcurrent}%
118            \xdef\NFLversionlengthmax{\filename@version}%
119           }%
120         \fi%
121         \ifx\filename@version\empty\relax%
122           \StrBehind[1]{\@NFLtempb}{ }[\@NFLtempc]%
123         \else%
124           \StrBehind[2]{\@NFLtempb}{ }[\@NFLtempc]%
125         \fi%
126         \StrLen{\@NFLtempc}[\NFLcaptionlengthcurrent]%
127         \ifnum \NFLcaptionlengthcurrent > \NFLcaptionlengthtmp \relax%
128           {\xdef\NFLcaptionlengthtmp{\NFLcaptionlengthcurrent}%
129            \xdef\NFLcaptionlengthmax{\@NFLtempc}%
130           }%
131         \fi%
132         }%
133       \message{^^J^^J%
134         ******************************************************************************^^J%
135         Package nicefilelist Info:^^J}%
136       \@ifundefined{hw@maxprintline}{%
137         \IfPackageLoadedTF{hardwrap}{%
138           \message{\space\string\hw@maxprintline\space unknown.^^J}%
139           }{\message{\space\string\hw@maxprintline\space %
140               unknown because not loading the hardwrap package.^^J}%
141            }%
142         \message{\space Now guessing it to be 79.}%
143         \global\newcount\hw@maxprintline%
144         \hw@maxprintline=79\relax% the default value
145         }{% was already defined by hardwrap package or manually
146          }%
147       \StrLen{\NFLbaselengthmax .\NFLextlengthmax\NFLspaceI %
148         yyyy-mm-dd\NFLspaceII \NFLversionlengthmax \NFLspaceIII %
149         \NFLcaptionlengthmax}[\@NFLtempc]%
150       \message{%
151         Longest base file-name: \NFLbaselengthmax\space - %
152          \NFLbaselengthtmp\space characters^^J%
153         Longest file extension: \NFLextlengthmax\space - %
```

```
154          \NFLextlengthtmp\space characters^^J%
155         Longest file version: \space\space\NFLversionlengthmax\space - %
156          \NFLversionlengthtmp\space characters^^J%
157         Longest file caption: \space\space\NFLcaptionlengthmax\space - %
158          \NFLcaptionlengthtmp\space characters^^J%
159         Total length of^^J%
160          base-filename.extension\string\NFLspaceI\space date%
161           \string\NFLspaceII\space version\string\NFLspaceIII\space %
162           caption:^^J
163          \@NFLtempc\space characters^^J%
164         Linewidth: \the\hw@maxprintline\space characters^^J%
165         }%
```

There exists “beamercolorthememetropolis-highcontrast.sty”.

```
166       \ifnum \@NFLtempc > \the\hw@maxprintline\relax%
167         \IfPackageLoadedTF{hardwrap}{%
168            \xdef\roomforcaption{%
169              \the\numexpr(\the\hw@maxprintline-(\@NFLtempc-\NFLcaptionlengthtmp))\relax}%
170            \ifnum\numexpr(2*\roomforcaption) < \NFLcaptionlengthtmp\relax%
171              \xdef\NFLmaybe{%
172               \the\numexpr(\NFLbaselengthtmp+(2*\roomforcaption-\NFLcaptionlengthtmp)+1)\relax}%
173              \gdef\NFLbaselengthmaybe{}%
174              \newcounter{NFLcounter}%
175              \loop%
176                \xdef\NFLbaselengthmaybe{A\NFLbaselengthmaybe}%
177                \addtocounter{NFLcounter}{1}%
178                \ifnum\value{NFLcounter}<\NFLmaybe%
179              \repeat%
180              \message{Warning:^^J%
181                       Caption needs more than two lines, %
182                       wrapping was not implemented for this.^^J%
183                       Maybe set the lengths for base file-name, %
184                       -extension, and -version manually^^J%
185                       or increase the max_print_line.^^J%
186                       Instead of using option autolength, after^^J%
187                       \string\usepackage[<options>]{nicefilelist}^^J%
188                       try adding^^J%
189                       \string\MFfieldtemplate{f-base}{\NFLbaselengthmaybe}^^J%
190                       \string\MFfieldtemplate{f-ext}{\NFLextlengthmax}^^J%
191                       \string\MFfieldtemplate{f-version}{\NFLversionlengthmax}^^J%
192                       and recompile twice - %
193                       or look for the second longest file name^^J%
194                       and use that for \string\MFfieldtemplate{f-base}.^^J}%
195            \fi%
196           }{\message{Warning:^^J%
197                      File information does not always fit into a single %
198                      line; this might look ugly.^^J%
199                      Maybe set the lengths for base file-name, %
200                      -extension, and -version manually^^J%
201                      or increase the max_print_line.^^J}%
```

```
202           }%
203       \fi%
204       \message{%
205       ********************************************************************************^^J^^J}%
206       \if@filesw%
207         \immediate\write\@auxout{\string\IfPackageLoadedTF{nicefilelist}{\string\relax}{%
208                                  \string\newcommand\string\MFfieldtemplate[2]{\string\relax}}}%
209         \immediate\write\@auxout{\string\MFfieldtemplate{f-base}{\NFLbaselengthmax}}%
210         \immediate\write\@auxout{\string\MFfieldtemplate{f-ext}{\NFLextlengthmax}}%
211         \immediate\write\@auxout{\string\MFfieldtemplate{f-version}{\NFLversionlengthmax}}%
212       \fi%
213       }
214   \fi
```

2.5 Safe Tests

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

```
215   \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`".

2.6 Redefining `\listfiles`

Similarly to original LATEX, `\listfiles` carries almost everything that is needed for the file list only. 2012-10-29: little point in this, perhaps, in that the package should be loaded when running `\listfiles` is intended—TO-DO. Or maybe it is loaded *just in case?*

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

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

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

... postponed for future version...

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

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

"Title:"

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

This starts the loop through the list of files

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

Like LaTeX's `\reserved@b`:

```
226         \expandafter\let\expandafter\@tempb
227              \csname ver@\filename@base.\filename@ext\endcsname
```

Packages that `\relax` their `\ver@`... (e.g., fontenc) can use `\ver@@`... to store the version information instead:

```
228         \ifx\@tempb\relax%
229           \expandafter\let\expandafter\@tempb
230             \csname ver@@\filename@base.\filename@ext\endcsname
231         \fi%
```

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

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

Actually I would like to be able to do even the file-name parsing in an expandable way — for all systems, `texsys.cfg`?! TO-DO

```
233         \NFL@filerow{%
```

Now all parsing and checking must be expandable.

```
234           \NFL@make@macro@arg\MFrightinfield\@tempa      {f-base}.%
235           \NFL@make@macro@arg\MFleftinfield \filename@ext{f-ext}%
236           \NFLspaceI
237           \NFL@ifx@kbl\@tempb\relax\NFLnotfound{%
238             \NFL@make@macro@arg\NFL@space@split\@tempb
239                                                 \NFL@maybe@three
240                                                 \NFL@date@or@rest
241           }%
242         }%
243       }%
```

The line of stars:

```
244       \typeout{                        %% trick v 2012/03/29
245                \MFrightinfield{********}{f-base}***^^J}%
```

TO-DO or more stars as with longnamefilelist?

```
246     }%
```

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*⟩:

```
\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.

```
\def\NFL@space@split##1{%
  \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.

```
\def\NFL@return@space@split##1 ##2\@nil##3\@nil@##4##5##6{%
  \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.

```
    {##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".

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

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

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

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

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

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

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

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

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

Since 2017-03-08 LaTeX accepts dates formatted as yyyy-mm-dd instead of yyyy/mm/dd. Currently both formats are valid. Therefore the nicefilelist package needs to also accept "-" (dash) instead of "/" (slash) as separator.

```
    \def\NFL@if@dateD##1{\NFL@dashes##1\NFL@xi xyzxyzxyzx\@nil}%
    \def\NFL@dashes##1##2##3##4##5##6##7##8{%
      \NFL@ifx@kbl##5-%
        {\NFL@ifx@kbl##8-\NFL@ten@only\NFL@false}%
        \NFL@false
      {##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. 2.5.

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

Finally checking digits:

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

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

```
    \def\NFL@digits##1{%
      \NFL@ifx@kbl##1\@nnil\NFL@true{%
        \NFL@if@digit@code##1<0\NFL@false{%
          \NFL@if@digit@code##1>9\NFL@false\NFL@digits
        }%
      }%
    }%
```

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

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

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

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

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

```
    \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:

```
288     \def\NFL@no@date@version{%
289       \NFLnodate\NFLspaceII\NFLnoversion@\NFLspaceIII}%
```

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

```
290     \def\NFLnoversion@{%
291       \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.

```
292     \def\NFL@maybe@three##1##2{%
293       \NFL@if@dateS{##1}%
294                    {##1\NFLspaceII
295                     \NFL@space@split{##2}%
296                                     \NFL@maybe@version@rest
297                                     \NFL@version@or@rest}%
298                    {\NFL@if@dateD{##1}%
299                                  {##1\NFLspaceII
300                                   \NFL@space@split{##2}%
301                                                   \NFL@maybe@version@rest
302                                                   \NFL@version@or@rest}%
303                    {\NFL@no@date@version##1 ##2}}%
```

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

```
304     \def\NFL@version@or@rest##1{%
305       \NFL@if@version{##1}%
306                      {\NFL@place@version{##1}}%
307                      {\NFLnoversion@\NFLspaceIII##1}}%
```

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

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

TO-DO: 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:

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

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

```
312           \@listfiles
313       \noexpand\@nil}%
314     \let\@listfiles\relax
```

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

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

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

```
316     \def\NFL@maybe@version@rest##1##2{%
317       \NFL@if@version{##1}%
318                      {\NFL@place@version{##1}\NFLspaceIII##2}%
319                      {\NFLnoversion@\NFLspaceIII##1 ##2}}%
320   }
```

2.7 Shorthand for **myfilist**

`\MaxBaseEmptyList{`⟨*longest-name*⟩`}[`⟨*read-again-files*⟩`]`

(v0.5) or

`\MaxBaseEmptyList*[`⟨*read-again-files*⟩`]`

(v0.6) as described in Section 1.5.2:

```
321   \newcommand*{\MaxBaseEmptyList}{%
322       \@ifstar{\maxBaseEmptyList{abcdabcd}}\maxBaseEmptyList}
323   \newcommand*{\maxBaseEmptyList}[1]{%
324       \MFfieldtemplate{f-base}{#1}%
325       \RequirePackage{myfilist}\EmptyFileList}
```

So `\maxBaseEmptyList` is like former `\MaxBaseEmptyList` without expecting a star—available to users.

2.8 Leaving the Package File

```
326   \endinput
```

2.9 VERSION HISTORY

```
327   v0.1   2012/03/20   started
328          2012/03/22   almost ready
329          2012/03/23   debugging; \NFLspaceI etc.;
330                       documentation completed
331   v0.2   2012/03/24   file info processed by \typeout - start
332          2012/03/25   trying, debugging
333          2012/03/26   continued; \NFL@place@version, \NFLnoversion@;
334                       works, reordered; another fix about Q -> \@empty
335          2012/03/27   undone the latter, explained; improved remarks on
336                       \@listfiles
337          2012/03/29   alignment of title/stars with base<11
338   v0.30  2012/05/18f. \NFL@ifx@kbl in \NFL@return@space@split
339          2012/05/20   all \ifx reimplemented, old code kept
340          STORED INTERNALLY
341   v0.31  2012/05/20   removing old code - STORED INTERNALLY
342   v0.32  2012/05/20   removing \NFL@xpxpxp; replacing \NFL@after@false
```

```
343                     by \NFL@ifnum@kbl, keeping old code
344          STORED INTERNALLY
345   v0.33  2012/05/20   removing old code; added 3 %s
346          STORED INTERNALLY
347   v0.4   2012/05/20   option [r]
348   v0.5   2012/09/30   \MaxBaseEmptyList
349   v0.6   2012/10/03   \MaxBaseEmptyLists first arg. only optional
350          2012/10/11   ... bad with 2nd opt. arg., *
351   v0.7   2012/10/13   "updating" date in \Provides...!
352          2012/10/28   \HardWrap first try
353          2012/10/29   \HardWrap newline material -> [wrap]
354                       sec:test below sec:opt, mentioning 'url'
355          2012/10/30   correcting \NFL@filerow without wrapping,
356                       doc.: |...| in sec:opt
357   v0.7a  2012/12/12   doc. monospace -> monofill; archived at:
358                       https://web.archive.org/web/20221205210517/
359                        https://mirror.mwt.me/ctan/install/macros/
360                        latex/contrib/nicefilelist.tds.zip
361   v0.8a  2022/12/05   Accepting also dashes instead of slashes in date
362                       (one-time fix by H.-Martin Münch); archived at:
363                       https://web.archive.org/web/20230106193203/
364                         https://mirror.mwt.me/ctan/install/macros/
365                         latex/contrib/nicefilelist.tds.zip
366   v0.9a  2023/01/08   now also regarding |ver@@| for version;
367                       new option [autolength] (using .aux file)
368                       (one-time fix by H.-Martin Münch)
369   v0.9b  2023/02/13   bug-fix: file extension missed when |\input|
370                       (one-time fix by H.-Martin Münch)
371
```

3 Credits

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

2. For Alois Kabelschacht—whose idea in TUGboat **8** #2[1] is used for v0.3—cf. the dowith documentation.

3. Another idea from Martin Münch: wrapping inside caption column. Implemented by UL with help of hardwrap as option `wrap`.

4 Missing

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

[1] "`\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`)