nicefilelist.sty

—

`\listfiles` Alignment for Connoisseurs*

Uwe Lück†

October 30, 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.

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 v0.7 of nicefilelist.sty as of 2012/10/30.

†`http://contact-ednotes.sty.de.vu`

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

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`, or `r,wrap` ...—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`' 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`. 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.

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*
  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
```

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  2012/10/30  r0.7   v0.7 [wrap] option
-----PACKAGE.---   --  -- --   --     --
nicefilelist.sty  2012/10/30  v0.7   more file list alignment (UL)
------DOCSRC.---   --  -- --   --     --
nicefilelist.tex  2012/10/30   --    documenting nicefilelist.sty
    srcfiles.tex  2012/10/30   --    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  2012/10/25  v0.7   \listfiles -- mine only (UL)
    readprov.sty  2012/03/20  v0.3b  file infos without loading (UL)
    fifinddo.sty  2012/08/27  v0.6   filtering TeX(t) files by TeX (UL)
     makedoc.sty  2012/08/28  v0.52  TeX input from *.sty (UL)
    niceverb.sty  2012/09/27  v0.5   minimize doc markup (UL)
    texlinks.sty  2012/05/13  v0.6   TeX-related links (UL)
     makedoc.cfg  2012/10/29   --    documentation settings
    mdoccorr.cfg  2011/12/03   --    makedoc local typographical corrections
-not-so-much.---   --  -- --   --     --
   kvsetkeys.sty  2009/07/30  v1.5   Key value parser with default handler
                                     support (HO)

     ***********

 List made at 2012/10/30, 20:12
 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}
             [2012/10/30 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)

```
1  \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2  \ProvidesPackage{nicefilelist}[2012/10/30 v0.7
3                                 more file list alignment (UL)]
4  %% Copyright (C) 2012 Uwe Lueck,
5  %% http://www.contact-ednotes.sty.de.vu
6  %% -- author-maintained in the sense of LPPL below --
7  %%
8  %% This file can be redistributed and/or modified under
9  %% the terms of the LaTeX Project Public License; either
10 %% version 1.3c of the License, or any later version.
11 %% The latest version of this license is in
12 %%     http://www.latex-project.org/lppl.txt
13 %% We did our best to help you, but there is NO WARRANTY.
14 %%
15 %% Please report bugs, problems, and suggestions via
16 %%
17 %%   http://www.contact-ednotes.sty.de.vu
18 %%
```

2.2 Alignment Settings

We use the monofill package for alignment of plain text:

```
19  \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 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.

```
20  \MFfieldtemplate{f-base}{nicefilelist}
21  \MFfieldtemplate{f-ext}{tex}
22  \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":

```
23  \newcommand*{\NFLspaceI}  { \space}
24  \newcommand*{\NFLspaceII} { \space}
25  \newcommand*{\NFLspaceIII}{ }
```

2.3 Failure Displays

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

```
26  \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:

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

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

```
28  \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. TODO: 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 automatical 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` does'nt work with myfilist's redefinition of `\typeout`. *With* `[wrap]`, `\NFL@filerow` applies hardwrap's `\HardWrap`:

```
\DeclareOption{wrap}{%
  \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 filenames with hardwrap seems to need

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

from monospace 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*⟩`}`

```
\ProcessOptions
```

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

```
\ifx\NFL@filerow\typeout \else
  \RequirePackage{hardwrap}
\fi
```

2.5 Safe Tests

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

```
\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—TODO. Or maybe it is loaded *just in case?*

```
\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`:

```
        \expandafter\let\expandafter\@tempb
            \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!

```
65          \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

```
66          \NFL@filerow{%
```

Now all parsing and checking must be expandable.

```
67            \NFL@make@macro@arg\MFrightinfield\@tempa      {f-base}.%
68            \NFL@make@macro@arg\MFleftinfield \filename@ext{f-ext}%
69            \NFLspaceI
70            \NFL@ifx@kbl\@tempb\relax\NFLnotfound{%
71              \NFL@make@macro@arg\NFL@space@split\@tempb
72                                                  \NFL@maybe@three
73                                                  \NFL@date@or@rest
74            }%
75          }%
76        }%
```

The line of stars:

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

TODO or more stars as with longnamefilelist?

```
79      }%
```

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*⟩:

```
80      \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.

```
81      \def\NFL@space@split##1{%
82        \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.

```
83      \def\NFL@return@space@split##1 ##2\@nil##3\@nil@##4##5##6{%
84        \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.

```
85      {##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."

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

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

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

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

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

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

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

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

```
96      {##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.

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

Finally checking digits:

```
99      ##1##2##3\@nnil}%
```

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

```
100   \def\NFL@digits##1{%
101     \NFL@ifx@kbl##1\@nnil\NFL@true{%
102       \NFL@if@digit@code##1<0\NFL@false{%
103         \NFL@if@digit@code##1>9\NFL@false\NFL@digits
104       }%
105     }%
106   }%
```

`\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:

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

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

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

```
\def\NFL@maybe@three##1##2{%
  \NFL@if@date{##1}%
              {##1\NFLspaceII
               \NFL@space@split{##2}%
                               \NFL@maybe@version@rest
                               \NFL@version@or@rest}%
              {\NFL@no@date@version##1 ##2}}%
```

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

```
\def\NFL@version@or@rest##1{%
  \NFL@if@version{##1}%
                 {\NFL@place@version{##1}}%
                 {\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 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:

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

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

2.8 Leaving the Package File

```
\endinput
```

2.9 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
       STORED INTERNALLY
v0.31  2012/05/20   removing old code - STORED INTERNALLY
v0.32  2012/05/20   removing \NFL@xpxpxp; replacing \NFL@after@false
                    by \NFL@ifnum@kbl, keeping old code
       STORED INTERNALLY
v0.33  2012/05/20   removing old code; added 3 %s
       STORED INTERNALLY
v0.4   2012/05/20   option [r]
v0.5   2012/09/30   \MaxBaseEmptyList
v0.6   2012/10/03   \MaxBaseEmptyLists first arg. only optional
       2012/10/11   ... bad with 2nd opt. arg., *
v0.7   2012/10/13   "updating" date in \Provides...!
       2012/10/28   \HardWrap first try
       2012/10/29   \HardWrap newline material -> [wrap]
                    sec:test below sec:opt, mentioning 'url'
       2012/10/30   correcting \NFL@filerow without wrapping,
                    doc.: |...| in sec:opt
```

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[2] is used for v0.3—cf. the dowith documentation.

4 Missing

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

[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`)