nicefilelist.sty

—

`\listfiles` Alignment for Connoisseurs*

Uwe Lück

April 21, 2025

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" (do not 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 file names and captions can be quite long without disturbing the alignment.

v0.9a offers a package option `[autolength]` for automatic sizing of the columns, using the xstring package.

v0.9c offers the options `[hashes]` and/or `[sizes]` to the `\listfiles` command, `\listfiles[hashes,sizes]`, as provided by LaTeX release 2024-06-01, see LaTeX News Issue 39 at `https://www.latex-project.org/news/latex2e-news/ltnews39.pdf`.

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

1 Features and Usage **2**
- 1.1 Relation to longnamefilelist.sty 2
- 1.2 Installing 3
- 1.3 Calling 3
- 1.4 Choosing Settings 3
  - 1.4.1 The Columns, Their Widths, and Their "Missing" Content 3

*This document describes version v0.9c of nicefilelist.sty as of 2025/04/21.

1.4.2 The Caption Column 4
1.4.3 Extended information in `listfiles` 5
1.5 Usage and examples with myfilist.sty 5
1.5.1 Basically 5
1.5.2 More Generally and Shorthand 7
1.5.3 Sample with Wrapped Caption Column 7

2 Implementation **9**
2.1 Package File Header (Legalese) 9
2.2 Alignment Settings 10
2.3 Failure Displays 10
2.4 Package Options 11
2.5 Safe Tests 15
2.6 Redefining `\listfiles` 16
2.7 Shorthand for myfilist 26
2.8 Leaving the Package File 26
2.9 VERSION HISTORY 26

3 Credits **28**

4 Missing **28**

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 do not 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*⟩ does not 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 do not 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. 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 does not 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).

1.4.3 Extended information in `listfiles`

Since LaTeX release 2024-06-01 the `\listfiles` command takes the option `hashes` to give the MD5 hash and `sizes` to give the file size of each file in the `.log`, printed after the end of the caption (in a new line); see LaTeX News Issue 39 at `https://www.latex-project.org/news/latex2e-news/ltnews39.pdf`.

1.5 Usage and examples 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  2025/04/21  v0.9c  options sizes \& hashes for \listfiles
-----PACKAGE.---   --  -- --   --     --
nicefilelist.sty  2025/04/21  v0.9c  More file list alignment (UL; HMM)
------DOCSRC.---   --  -- --   --     --
nicefilelist.tex  2025/04/21   --    documenting nicefilelist.sty
    srcfiles.tex  2025/04/21   --    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  2012/11/27  v0.51  minimize doc markup (UL)
    texlinks.sty  2013/01/22  v0.82  TeX-related links (UL)
     makedoc.cfg  2012/11/30   --    documentation settings
    mdoccorr.cfg  2012/11/13   --    'makedoc' local typographical corrections
-not-so-much.---   --  -- --   --     --
   kvsetkeys.sty  2012/04/25  v1.16  Key value parser (HO)
     **********

 List made at 2025/04/21, 19:16
 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  2025/04/21  v0.9c  options sizes \& hashes for \listfiles
-----PACKAGE.---   --  -- --   --     --
nicefilelist.sty  2025/04/21  v0.9c  More file list alignment (UL; HMM)
------DOCSRC.---   --  -- --   --     --
nicefilelist.tex  2025/04/21   --    documenting nicefilelist.sty
    srcfiles.tex  2025/04/21   --    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  2012/11/27  v0.51  minimize doc markup (UL)
    texlinks.sty  2013/01/22  v0.82  TeX-related links (UL)
     makedoc.cfg  2012/11/30   --    documentation settings
    mdoccorr.cfg  2012/11/13   --    'makedoc' local typographical corrections
-not-so-much.---   --  -- --   --     --
   kvsetkeys.sty  2012/04/25  v1.16  Key value parser (HO)
     ***********

 List made at 2025/04/21, 19:16
 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}
             [2025/04/21 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]% Option autolength needs 2022-11-01, parameters for \listfiles
2  \ProvidesPackage{nicefilelist}[2025/04/21 v0.9c
3                                  More file list alignment (UL; HMM)]
4  %% Copyright (C) 2012 Uwe LÃck (deceased June 2020), 2022-2025 H.-Martin MÃnch
5  %%
6  %% This work may be distributed and/or modified under the
7  %% conditions of the LaTeX Project Public License, either
8  %% version 1.3c of this license or (at your option) any later
9  %% version. This version of this license is in
10 %%    https://www.latex-project.org/lppl/lppl-1-3c.txt
11 %% and the latest version of this license is in
12 %%    https://www.latex-project.org/lppl.txt
13 %% and version 1.3c or later is part of all distributions of
14 %% LaTeX version 2005-12-01 or later.
15 %%
16 %%
17 %%
18 %%
```

2.2 Alignment Settings

We use the monofill package for alignment of plain text:

```
19  \RequirePackage{monofill}[2012/10/29]% v0.2 monospace alignment (UL)
```

See its documentation for details.

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, and then issue `\listfiles` or load myfilist.sty.

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

`\NFLspaceI`, `\NFLspaceII`, `\NFLspaceIII`, and `NFLspaceIV` determine the space between the five columns for names, dates, versions, "captions" and sizes/hashes (when sizes/hashes have been requested by option):

```
23  \newcommand*{\NFLspaceI}  { \space}
24  \newcommand*{\NFLspaceII} { \space}
25  \newcommand*{\NFLspaceIII}{ }
26  \newcommand*{\NFLspaceIV} {^^J% <- newline
27    \ifNFLwrap\else%
28      \MFrightinfield\space{f-base} %
29      \MFleftinfield \space{f-ext}%
30      \NFLspaceI\@spaces\space\@spaces\space\NFLspaceII%
31      \MFrightinfield\space{f-version}\NFLspaceIII%
32    \fi%
33  }% original kernel: ^^J\@spaces
```

2.3 Failure Displays

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

```
34  \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:

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

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

```
36  \def\NFL@hashessizes{}
37  \newcommand*{\NFLnotfound}{ * NOT FOUND * \NFLspaceII\NFLspaceIII\space\NFL@hashessizes}
```

where `\NFL@hashessizes` is redefined later.

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 does not 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*⟩`}`

Since v0.9a the package option `[autolength]` is provided 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}{%
         Option autolength needs newer LaTeX format%
       }{Needed LaTeX format version: 2022-11-01 or newer.\MessageBreak%
         Found \space LaTeX format version: \fmtversion.\MessageBreak%
         To use option autolength update your TeX distribution.%
       }
     }
  }

\ProcessOptions

```

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

```
\ifNFLwrap\RequirePackage{hardwrap}[2011/02/12]% v0.2 Hard wrap messages (KG, WR)
\fi

\ifNFLautolength
  \RequirePackage{xstring}[2023-08-22]% v1.86 String manipulations (CT); requires e-TeX
  \AddToHook{enddocument/afterlastpage}{%
    \xdef\NFLbaselengthmax{nicefilelist}%
    \xdef\NFLextlengthmax{sty}%
    \xdef\NFLversionlengthmax{v0.9c}%
    \xdef\NFLcaptionlengthmax{More file list alignment (UL; HMM)}%
    \xdef\NFLbaselengthtmp{0}%
    \xdef\NFLextlengthtmp{0}%
```

```
89      \xdef\NFLversionlengthtmp{0}%
90      \xdef\NFLcaptionlengthtmp{0}%
91      \xdef\NFLtotallengthtmp{0}%
92      \@for\@currname:=\@filelist\do{% This starts the loop through the list of files.
93        \filename@parse\@currname%
94        \edef\filename@ext{\ifx\filename@ext\relax tex\else\filename@ext\fi}%
95        \StrLen{\filename@base}[\NFLbaselengthcurrent]%
96        \ifnum \NFLbaselengthcurrent > \NFLbaselengthtmp \relax%
97          {\xdef\NFLbaselengthtmp{\NFLbaselengthcurrent}%
98           \xdef\NFLbaselengthmax{\filename@base}%
99          }%
100       \fi%
101       \StrLen{\filename@ext}[\NFLextlengthcurrent]%
102       \ifnum \NFLextlengthcurrent > \NFLextlengthtmp \relax%
103         {\xdef\NFLextlengthtmp{\NFLextlengthcurrent}%
104          \xdef\NFLextlengthmax{\filename@ext}%
105         }%
106       \fi%
107       \expandafter\let\expandafter\@NFLtempb\csname ver@\filename@base.\filename@ext\endcsname%
108       \StrBetween[1,2]{\@NFLtempb}{ }{ }[\@NFLtempc]%
109       % assuming date space version space description
110       \IfBeginWith{\@NFLtempc}{v}{%
111         % fails if version omitted and description starts with v - evil!
112         \xdef\filename@version{\@NFLtempc}}{%
113         % else: empty or unusual format
114         % Packages that \relax their \ver@... (e.g., fontenc)
115         % can use \ver@@... to store the version information instead:
116         \expandafter\let\expandafter\@NFLtempd\csname ver@@\filename@base.\filename@ext\endcsname%
117         \StrBetween[1,2]{\@NFLtempd}{ }{ }[\@NFLtempc]%
118         % assuming date space version space description again
119         \IfBeginWith{\@NFLtempc}{v}{\xdef\filename@version{\@NFLtempc}%
120          }{% else: no usual version found
121            \xdef\filename@version{}%
122           }%
123         }%
124       \StrLen{\filename@version}[\NFLversionlengthcurrent]%
125       \ifnum\NFLversionlengthcurrent > \NFLversionlengthtmp\relax%
126         {\xdef\NFLversionlengthtmp{\NFLversionlengthcurrent}%
127          \xdef\NFLversionlengthmax{\filename@version}%
128         }%
129       \fi%
130       \ifx\filename@version\empty\relax%
131         \StrBehind[1]{\@NFLtempb}{ }[\@NFLtempc]%
132       \else%
133         \StrBehind[2]{\@NFLtempb}{ }[\@NFLtempc]%
134       \fi%
135       \StrLen{\@NFLtempc}[\NFLcaptionlengthcurrent]%
136       \ifnum\NFLcaptionlengthcurrent > \NFLcaptionlengthtmp\relax%
137         {\xdef\NFLcaptionlengthtmp{\NFLcaptionlengthcurrent}%
138          \xdef\NFLcaptionlengthmax{\@NFLtempc}%
```

```
      }%
    \fi%
    }%
  \message{^^J^^J%
    ******************************************************************************^^J%
    Package nicefilelist Info:^^J}%
  \@ifundefined{hw@maxprintline}{%
    \IfPackageLoadedTF{hardwrap}{%
      \message{\space\string\hw@maxprintline\space unknown.^^J}%
      }{\message{\space\string\hw@maxprintline\space %
          unknown because not loading the hardwrap package.^^J}%
       }%
    \message{\space Now guessing it to be 79.}%
    \global\newcount\hw@maxprintline%
    \hw@maxprintline=79\relax% the default value
    }{% was already defined by hardwrap package or manually
     }%
  \StrLen{\NFLbaselengthmax .\NFLextlengthmax\NFLspaceI %
    yyyy-mm-dd\NFLspaceII \NFLversionlengthmax \NFLspaceIII %
    \NFLcaptionlengthmax}[\@NFLtempc]%
  \message{%
    Longest base file-name: \NFLbaselengthmax\space - %
     \NFLbaselengthtmp\space characters^^J%
    Longest file extension: \NFLextlengthmax\space - %
     \NFLextlengthtmp\space characters^^J%
    Longest file version: \space\space\NFLversionlengthmax\space - %
     \NFLversionlengthtmp\space characters^^J%
    Longest file caption: \space\space\NFLcaptionlengthmax\space - %
     \NFLcaptionlengthtmp\space characters^^J%
    Total length of^^J%
     base-filename.extension\string\NFLspaceI\space date%
      \string\NFLspaceII\space version\string\NFLspaceIII\space %
      caption:^^J
     \@NFLtempc\space characters^^J%
    Linewidth: \the\hw@maxprintline\space characters^^J%
    }%
```

There exists “beamercolorthememetropolis-highcontrast.sty”.

```
  \ifnum \@NFLtempc > \the\hw@maxprintline\relax%
    \IfPackageLoadedTF{hardwrap}{%
       \xdef\roomforcaption{%
         \the\numexpr(\the\hw@maxprintline-(\@NFLtempc-\NFLcaptionlengthtmp))\relax}%
       \ifnum\numexpr(2*\roomforcaption) < \NFLcaptionlengthtmp\relax%
         \xdef\NFLmaybe{%
          \the\numexpr(\NFLbaselengthtmp+(2*\roomforcaption-\NFLcaptionlengthtmp)+1)\relax}%
         \gdef\NFLbaselengthmaybe{}%
         \newcounter{NFLcounter}%
         \loop%
           \xdef\NFLbaselengthmaybe{A\NFLbaselengthmaybe}%
           \addtocounter{NFLcounter}{1}%
```

```
187               \ifnum\value{NFLcounter}<\NFLmaybe%
188             \repeat%
189             \message{Warning:^^J%
190                     Caption needs more than two lines, %
191                     wrapping was not implemented for this.^^J%
192                     Maybe set the lengths for base file-name, %
193                     -extension, and -version manually^^J%
194                     or increase the max_print_line.^^J%
195                     Instead of using option autolength, after^^J%
196                     \string\usepackage[<options>]{nicefilelist}^^J%
197                     try adding^^J%
198                     \string\MFfieldtemplate{f-base}{\NFLbaselengthmaybe}^^J%
199                     \string\MFfieldtemplate{f-ext}{\NFLextlengthmax}^^J%
200                     \string\MFfieldtemplate{f-version}{\NFLversionlengthmax}^^J%
201                     and recompile twice - %
202                     or look for the second longest file name^^J%
203                     and use that for \string\MFfieldtemplate{f-base}.^^J}%
204           \fi%
205         }{\message{Warning:^^J%
206                   File information does not always fit into a single %
207                   line; this might look ugly.^^J%
208                   Maybe set the lengths for base file-name, %
209                   -extension, and -version manually^^J%
210                   or increase the max_print_line.^^J}%
211          }%
212       \fi%
213       \message{%
214       ********************************************************************************^^J^^J}%
215       \if@filesw%
216         \immediate\write\@auxout{\string\IfPackageLoadedTF{nicefilelist}{\string\relax}{%
217                               \string\newcommand\string\MFfieldtemplate[2]{\string\relax}}}%
218         \immediate\write\@auxout{\string\MFfieldtemplate{f-base}{\NFLbaselengthmax}}%
219         \immediate\write\@auxout{\string\MFfieldtemplate{f-ext}{\NFLextlengthmax}}%
220         \immediate\write\@auxout{\string\MFfieldtemplate{f-version}{\NFLversionlengthmax}}%
221       \fi%
222       }
223   \fi
```

2.5 Safe Tests

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

```
224   \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?*

```
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}

\IfFormatAtLeastTF{2024/06/01}{% new format:
  \RenewDocumentCommand\listfiles{O{}}{%
    \SetKeys[__kernel/listfiles]{#1}%
    \let\listfiles\relax%
```

– this clears memory. Now LaTeX does not collect file names for \listfiles when \@listfiles is undefined, therefore

```
    \def\NFL@listfiles##1##2##3##4##5##6##7##8##9\@@{%
    \def\reserved@d{\\}%
    \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
    \ifx\reserved@c\reserved@d
      \edef\filename@area{ \filename@area}%
    \fi}}%
```

\@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
             \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
```

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

```
      \ifx\@tempb\relax%
        \expandafter\let\expandafter\@tempb
           \csname ver@@\filename@base.\filename@ext\endcsname
      \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!

```
250         \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

```
251         \NFL@filerow{%
```

Now all parsing and checking must be expandable.

```
252           \NFL@make@macro@arg\MFrightinfield\@tempa      {f-base}.%
253           \NFL@make@macro@arg\MFleftinfield \filename@ext{f-ext}%
254           \NFLspaceI
255           \NFL@ifx@kbl\@tempb\relax\NFLnotfound{%
256             \NFL@make@macro@arg\NFL@space@split\@tempb
257                                                \NFL@maybe@three
258                                                \NFL@date@or@rest
259           }%
260         }%
261       }%
```

The line of stars:

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

TO-DO or more stars as with longnamefilelist?

```
264     }%
```

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*⟩:

```
265     \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.

```
266     \def\NFL@space@split##1{%
267       \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.

```
268     \def\NFL@return@space@split##1 ##2\@nil##3\@nil@##4##5##6{%
269       \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 do not 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@dateS{##1}%
                   {##1\NFLspaceII
                    \NFL@space@split{##2}%
                                    \NFL@maybe@version@rest
                                    \NFL@version@or@rest}%
                   {\NFL@if@dateD{##1}%
                                 {##1\NFLspaceII
                                  \NFL@space@split{##2}%
                                                  \NFL@maybe@version@rest
                                                  \NFL@version@or@rest}%
                   {\NFL@no@date@version##1 ##2\NFL@hashessizes}}}%
```

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

```
\def\NFL@version@or@rest##1{%
  \NFL@if@version{##1}%
                 {\NFL@place@version{##1}}%
                 {\begingroup%
                    \toks0={##1}%
                    \edef\NFLcap{\the\toks0}%
                  \expandafter\endgroup%
                  \ifx\NFLcap\empty%
                  \else%
                    \NFLnoversion@\NFLspaceIII##1%
                  \fi%
                 }}%
```

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

```
\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:

```
\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@hashessizes` adds the sizes and/or hashes, when the respective optional option to `\listfiles` has been used, e.g. `\listfiles[sizes,hashes]`:

```
\def\NFL@hashessizes{%
  \ifnum0%
    \if@listfiles@hashes1\fi%
    \if@listfiles@sizes1\fi%
    >0 %
    \NFLspaceIV(%
      \if@listfiles@sizes%
        size \@dofilelist@size\@currname%
        \if@listfiles@hashes%
          , %
```

```
            \fi%
          \fi%
          \if@listfiles@hashes%
            hash \@dofilelist@hash\@currname%
          \fi%
        )%
      \fi%
    }
```

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

```
    \def\NFL@maybe@version@rest##1##2{%
      \NFL@if@version{##1}%
        {\NFL@place@version{##1}\NFLspaceIII}%
        {\NFLnoversion@\NFLspaceIII##1 }%
      ##2\NFL@hashessizes}%
    }%
  }{% older format:
    \renewcommand*{\listfiles}{%
  %
      \let\listfiles\relax%
```

– this clears memory. Now LaTeX does not collect file names for `\listfiles` when `\@listfiles` is undefined, therefore

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

... postponed for future version... `\@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
               \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
```

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

```
380         \ifx\@tempb\relax%
381           \expandafter\let\expandafter\@tempb
382              \csname ver@@\filename@base.\filename@ext\endcsname
383         \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!

```
384         \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

```
385         \NFL@filerow{%
```

Now all parsing and checking must be expandable.

```
386           \NFL@make@macro@arg\MFrightinfield\@tempa      {f-base}.%
387           \NFL@make@macro@arg\MFleftinfield \filename@ext{f-ext}%
388           \NFLspaceI
389           \NFL@ifx@kbl\@tempb\relax\NFLnotfound{%
390             \NFL@make@macro@arg\NFL@space@split\@tempb
391                                                 \NFL@maybe@three
392                                                 \NFL@date@or@rest
393           }%
394         }%
395       }%
```

The line of stars:

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

TO-DO or more stars as with longnamefilelist?

```
398     }%
```

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*⟩:

```
399     \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.

```
400     \def\NFL@space@split##1{%
401       \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 do not 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.

```
444     \def\NFL@maybe@three##1##2{%
445       \NFL@if@dateS{##1}%
446                    {##1\NFLspaceII
447                     \NFL@space@split{##2}%
448                                     \NFL@maybe@version@rest
449                                     \NFL@version@or@rest}%
450                    {\NFL@if@dateD{##1}%
451                                  {##1\NFLspaceII
452                                   \NFL@space@split{##2}%
453                                                   \NFL@maybe@version@rest
454                                                   \NFL@version@or@rest}%
455                    {\NFL@no@date@version##1 ##2\NFL@hashessizes}}}%
```

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

```
456     \def\NFL@version@or@rest##1{%
457       \NFL@if@version{##1}%
458                      {\NFL@place@version{##1}}%
459                      {\begingroup%
460                         \toks0={##1}%
461                         \edef\NFLcap{\the\toks0}%
462                       \expandafter\endgroup%
463                       \ifx\NFLcap\empty%
464                       \else%
465                         \NFLnoversion@\NFLspaceIII##1%
466                       \fi%
467                      }}%
```

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

```
468     \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:

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

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

```
472           \@listfiles
473       \noexpand\@nil}%
474     \let\@listfiles\relax
```

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

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

`\NFL@hashessizes` adds the sizes and/or hashes, when the respective optional option to `\listfiles` has been used, e.g. `\listfiles[sizes,hashes]`:

```
    \def\NFL@hashessizes{%
      % is not implemented for old formats
    }
```

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

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

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
```

```
501         2012/05/20   all \ifx reimplemented, old code kept
502         STORED INTERNALLY
503   v0.31 2012/05/20   removing old code - STORED INTERNALLY
504   v0.32 2012/05/20   removing \NFL@xpxpxp;
505                      replacing \NFL@after@false by \NFL@ifnum@kbl, keeping old code
506         STORED INTERNALLY
507   v0.33 2012/05/20   removing old code; added 3 %s
508         STORED INTERNALLY
509   v0.4  2012/05/20   option [r]
510   v0.5  2012/09/30   \MaxBaseEmptyList
511   v0.6  2012/10/03   \MaxBaseEmptyLists first arg. only optional
512         2012/10/11   ... bad with 2nd opt. arg., *
513   v0.7  2012/10/13   "updating" date in \Provides...!
514         2012/10/28   \HardWrap first try
515         2012/10/29   \HardWrap newline material -> [wrap]
516                      sec:test below sec:opt, mentioning 'url'
517         2012/10/30   correcting \NFL@filerow without wrapping, doc.: |...| in sec:opt
518   v0.7a 2012/12/12   doc. monospace -> monofill; archived at:
519                      https://web.archive.org/web/20221205210517/
520                       https://mirror.mwt.me/ctan/install/macros/
521                       latex/contrib/nicefilelist.tds.zip
522   v0.8a 2022/12/05   Accepting also dashes instead of slashes in date
523                      (one-time fix by H.-Martin M\"{u}nch); archived at:
524                      https://web.archive.org/web/20230106193203/
525                        https://mirror.mwt.me/ctan/install/macros/
526                        latex/contrib/nicefilelist.tds.zip
527   v0.9a 2023/01/08   now also regarding |ver@@| for version;
528                      new option [autolength] (using .aux file)
529                      (one-time fix by H.-Martin M\"{u}nch)
530   v0.9b 2023/02/13   bug-fix: file extension missed when |\input|
531                      (one-time fix by H.-Martin M\"{u}nch)
532   v0.9c 2025/04/21   kernel's |\listfiles| now accepts options "sizes" and "hashes"
533                      and displays file size and md5 hash -
534                      now also implemented for nicefilelist
535                      (one-time fix by H.-Martin M\"{u}nch)
536
```

3 Credits

1. It was H.-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 H.-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.

Well, nicefilelist 2025/04/21 v0.9c *does* introduce the options `sizes` and `hashes` for `\listfiles`, because the recent kernel provides those! (Works only for formats 2024-06-01 and newer.)

[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`)