

# nicefilelist.sty

## `\listfiles` Alignment for Connoisseurs\*

Uwe Lück<sup>†</sup>

October 11, 2012

### Abstract

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

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

## Contents

<b>1</b>	<b>Features and Usage</b>	<b>2</b>
1.1	Relation to <code>longnamefilelist.sty</code> . . . . .	2
1.2	Installing . . . . .	2
1.3	Calling . . . . .	2
1.4	Usage and Sample with <code>myfilist.sty</code> . . . . .	3
1.4.1	Basically . . . . .	3
1.4.2	More Generally and Shorthand . . . . .	4
<b>2</b>	<b>Implementation</b>	<b>5</b>
2.1	Package File Header (Legalese) . . . . .	5
2.2	Alignment Settings . . . . .	5
2.3	Failure Displays . . . . .	6
2.4	Safe Tests . . . . .	6
2.5	Package Option <code>[r]</code> . . . . .	6
2.6	Redefining <code>\listfiles</code> . . . . .	7
2.7	Shorthand for <code>myfilist</code> . . . . .	11

---

\*This document describes version **v0.6** of `nicefilelist.sty` as of 2012/10/03.

<sup>†</sup><http://contact-ednotes.sty.de.vu>

1	FEATURES AND USAGE	2
2.8	Leaving the Package File . . . . .	11
2.9	VERSION HISTORY . . . . .	11
3	Credits	13
4	Missing	13

# 1 Features and Usage

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

## 1.1 Relation to `longnamefilelist.sty`

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

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

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

## 1.2 Installing

The file `nicefilelist.sty` is provided ready, installation only requires putting it somewhere where T<sub>E</sub>X finds it (which may need updating the filename data base).<sup>1</sup>

## 1.3 Calling

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

```
\usepackage{nicefilelist}
```

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

```
\RequirePackage{nicefilelist}
```

---

<sup>1</sup><http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf>

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

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

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

or

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

See Section 2.5 for more information.

## 1.4 Usage and Sample with `myfilist.sty`

### 1.4.1 Basically

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

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

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

The previous code mentions the following files:

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

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

`empty.f` just is an empty file.

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

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

```

*File List*
nicefilelist.sty 2012/03/23 v0.1 more file list alignment (UL)
monofill.sty    2012/03/19 v0.1a monospace alignment (UL)
myfilist.sty    2011/01/30 v0.3a \listfiles -- mine only (UL)
readprov.sty    2010/11/27 v0.3 file infos without loading (UL)
nicefilelist.tex 2012/03/23 -- documenting nicefilelist.sty
proonly.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

```

#### 1.4.2 More Generally and Shorthand

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

```

\RequirePackage[r]{nicefilelist}
\MFfieldtemplate{f-base}{\langle longest-name \rangle}
\RequirePackage{myfilist}
\EmptyFileList[\langle read-again-files \rangle]

```

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

```
\MFfieldtemplate{f-base}{\langle longest-name \rangle}
```

according to Section 2.2. While there are also field identifiers `f-ext` and `f-version`, it seems that only `f-base` varies between real-life applications.

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

```
\MaxBaseEmptyList{\langle longest-name \rangle}[\langle read-again-files \rangle]
```

—“optionally” without ‘`[\langle read-again-files \rangle]`’. 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 L<sup>A</sup>T<sub>E</sub>X’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`.

## 2 Implementation

### 2.1 Package File Header (Legalese)

```
1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{nicefilelist}[2012/10/03 v0.6
3                               more file list alignment (UL)]
4
5 %% Copyright (C) 2012 Uwe Lueck,
6 %% http://www.contact-ednotes.sty.de.vu
7 %% -- author-maintained in the sense of LPPL below --
8 %%
9 %% This file can be redistributed and/or modified under
10 %% the terms of the LaTeX Project Public License; either
11 %% version 1.3c of the License, or any later version.
12 %% The latest version of this license is in
13 %% http://www.latex-project.org/lppl.txt
14 %% We did our best to help you, but there is NO WARRANTY.
15 %%
16 %% Please report bugs, problems, and suggestions via
17 %%
18 %% http://www.contact-ednotes.sty.de.vu
19 %%
```

### 2.2 Alignment Settings

We use the `monofill` package for alignment of plain text:

```
20 \RequirePackage{monofill}
```

See its documentation for details.

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

```
21 \MFfieldtemplate{f-base}{nicefilelist}
22 \MFfieldtemplate{f-ext}{tex}
23 \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”:

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

## 2.3 Failure Displays

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

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

`\NFLnversion` 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:

```
28 \newcommand*\NFLnversion{\space--}
```

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

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

## 2.4 Safe Tests

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

```
30 \catcode'\Q=7 \let\nfl@criterion=Q \catcode'\Q=11
```

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

## 2.5 Package Option [r]

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

```
31 \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!?)

```

32 \DeclareOption{r}{%
33   \def\@listfiles{%
34     {\noexpand\NFL@ifx@kbl##1r%
35      {\noexpand\NFL@digits##2\noexpand\@nnil}%
36      \noexpand\NFL@false}%
37   }%
38 }
39 \ProcessOptions

```

## 2.6 Redefining \listfiles

Similarly to original L<sup>A</sup>T<sub>E</sub>X, `\listfiles` carries almost everything that is needed for the file list only:

```

40 \renewcommand*\listfiles{%
41   \let\listfiles\relax

```

—this clears memory. Now L<sup>A</sup>T<sub>E</sub>X doesn’t collect file names for `\listfiles` when `\@listfiles` is undefined, therefore

```

42 % \let\@listfiles\relax

```

... postponed for v0.4 ...

`\@dofilelist` is executed by the standard L<sup>A</sup>T<sub>E</sub>X `\enddocument` macro or by `\ListInfos` from the `myfilist` package.

```

43 \def\@dofilelist{%
44   "Title:"
45   \typeout{^^J          %% trick 2012/03/29 vv
46   \MFrightinfield{*File Lis}{f-base}t*}%
47   \for\@currname:=\@filelist\do{%

```

This starts the loop through the list of files

```

47   \filename@parse\@currname
48   \edef\filename@ext{%
49   \ifx\filename@ext\relax tex\else\filename@ext\fi}%

```

Like L<sup>A</sup>T<sub>E</sub>X’s `\reserved@b`:

```

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

```

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

```

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

```

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

53           \typeout{%

Now all parsing and checking must be expandable.

```

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

```

The line of stars:

```

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

```

**TODO** or more stars as with `longnamefilelist`?

```

66           }%

```

This finishes the definition of `\@dofilelist`.

`\NFL@make@macro@arg⟨cmd-1⟩⟨cmd-2⟩`

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

```

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

```

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

```

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

```

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

```

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

```

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

```

72   {##6{##4}}{##5{##1}{##2}}}%

```



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

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

Dealing with `\NFL@date@or@rest{<token-list>}` before `\NFL@maybe@three:`

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

`\NFL@if@date{<token-list>}{<yes>}{<no>}` ...

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

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

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

This especially happens when `<token-list>` is empty. Digit candidates back:

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

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

`\NFL@ten@only{<digits>}{<digit>}{<digit>Q}`

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

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

Finally checking digits:

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

`\NFL@digits<token>` is a loop through single tokens:

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

`\NFL@if@digit@code<char-1>{<relation>}{<char-2>}{<fits>}{<bad>}`:

```

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

```

`\NFL@false` skips further candidates and dummies and chooses *<no>*:

```

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

```

`\NFL@true` skips further candidates and dummies and chooses *<yes>*:

```

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

```

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

```

99 \def\nfl@no@date@version{%
100 \NFLnodate\nFLspaceII\nFLnversion@\nFLspaceIII}%

```

`\NFLnversion@` adds filler to `\NFLnversion`:

```

101 \def\nflnversion@{%
102 \NFL@make@macro@arg\nFL@place@version\nFLnversion}%

```

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

```

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

```

`\NFL@version@or@rest{<token-list>}`:

```

110 \def\nfl@version@or@rest##1{%
111 \NFL@if@version{##1}%
112 {\NFL@place@version{##1}}%
113 {\NFLnversion@\nFLspaceIII##1}}%

```

`\NFL@if@version{<token-list>}{<yes>}{<no>}`:

```

114 \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:

```

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

```

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

```

118 \@listfiles
119 \noexpand\@nil}%
120 \let\@listfiles\relax

```

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

```

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

122 \NFL@maybe@version@rest{<list-1>}{<list-2>}:
123 \def\NFL@maybe@version@rest##1##2{%
124 \NFL@if@version{##1}%
125 {\NFL@place@version{##1}\NFLspaceIII##2}%
126 {\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 1.4.2:

```

127 \newcommand*{\MaxBaseEmptyList}{%
128 \@ifstar{\maxBaseEmptyList{abcdabcd}}{\maxBaseEmptyList}
129 \newcommand*{\maxBaseEmptyList}[1]{%
130 \MFfieldtemplate{f-base}{#1}%
131 \RequirePackage{myfilist}\EmptyFileList}

```

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

## 2.8 Leaving the Package File

```

132 \endinput

```

## 2.9 VERSION HISTORY

```

133 v0.1    2012/03/20    started
134         2012/03/22    almost ready
135         2012/03/23    debugging; \NFLspaceI etc.;
136                     documentation completed
137

```

```

138 v0.2 2012/03/24 file info processed by \typeout - start
139      2012/03/25 trying, debugging
140      2012/03/26 continued; \NFL@place@version, \NFLnversion@;
141                  works, reordered; another fix about Q -> \@empty
142      2012/03/27 undone the latter, explained; improved remarks on
143                  \@listfiles
144      2012/03/29 alignment of title/stars with base<11
145
146 v0.30 2012/05/18f. \NFL@ifx@kbl in \NFL@return@space@split
147      2012/05/20 all \ifx reimplemented, old code kept
148      STORED INTERNALLY
149 v0.31 2012/05/20 removing old code - STORED INTERNALLY
150 v0.32 2012/05/20 removing \NFL@xpxpxp; replacing \NFL@after@false
151                  by \NFL@ifnum@kbl, keeping old code
152      STORED INTERNALLY
153 v0.33 2012/05/20 removing old code; added 3 %s
154      STORED INTERNALLY
155 v0.4 2012/05/20 option [r]
156 v0.5 2012/09/30 \MaxBaseEmptyList
157 v0.6 2012/10/03 \MaxBaseEmptyLists first arg. only optional
158      2012/10/11 ... bad with 2nd opt. arg., *
159

```

### 3 Credits

1. It was MARTIN MUENCH who pointed out the shortcomings of `longname-filelist` that the present package addresses—thanks!
2. For ALOIS KABELSCHACHT—whose idea in TUGboat 8 #2<sup>2</sup> is used for v0.3—cf. the `dowith` documentation.

### 4 Missing

1. The package once might provide `keyval`-style optional arguments for `\listfiles` or even call `\listfiles` automatically with `keyval` package options.
2. Another idea from MARTIN MUENCH: wrapping inside caption column. Can `hardwrap` help?

---

<sup>2</sup>“`\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](http://tug.org/TUGboat/tb08-2/tb18kabel.pdf))