FiNK – the LaTeX 2ε **Fi**le **N**ame **K**eeper *

Didier Verna
mailto:didier@lrde.epita.fr
http://www.lrde.epita.fr/~didier/

June 11, 2007

1 Description

This package is a real fink indeed: it looks over your shoulder and keeps track of files `\input`'ed (the LaTeX way) or `\include`'ed in your document. You then have a permanent access to the directory, name and extension of the file currently being processed through several macros. Dis packache fas orichinally a hack dat I used somefere elss, but since it might be off a cheneral interest, I'fe decided to make it a separate fink...

2 User Interface

To use the package, simply say `\usepackage[`⟨*options*⟩`]{fink}` in the preamble of your document. This will do everything for you. Available options will be described when appropriate.

2.1 Retrieving the current file's name components

`\finkdir` `\finkbase` `\finkext`

The file currently being processed is described by the macros `\finkdir`, `\finkbase` and `\finkext` which expand (as you may have guessed) to the directory, base name (sans extension), and extension of the file.

`\finkfile` `\finkpath`

Additionally, the macro `\finkfile` is defined to be `\finkbase.\finkext` (as in previous versions), and the macro `\finkpath` (new in version 2.0) is defined to be `\finkdir\finkfile`. Feel free to use these macros in your sources.

2.2 Main file's name components

`maindir` `mainext`

Because there's no way TeX can give you back information about the file being processed (apart from its base name), *FiNK* provides the options `maindir` (defaults to `./`) and `mainext` (defaults to `tex`) for changing the directory and the extension

*This document describes *FiNK* 2.0, release date 2007/06/11.

of the main source file. For instance, suppose your source file is in `src/foo.ltx` and you are compiling in `pdf/`. You can then use the package as follows:

```
\usepackage[maindir=../src,mainext=ltx]{fink}
```

3 AUC-TEX support

AUC-TEX is a powerful major mode for editing TEX documents in Emacs or XEmacs. In particular, it provides automatic completion of macro names once they are known. ***FiNK*** supports AUC-TEX by providing a style file named `fink.el` which contains AUC-TEX definitions for the relevant macros. This file should be installed to a location where AUC-TEX can find it (usually in a subdirectory of your LATEX styles directory). Please refer to the AUC-TEX documentation for more information on this.

4 Caveat

FiNK cannot follow files included with the TEX `\input` primitive. That's because TEX has a very insensible way of defining primitives whose argument parsing syntax is not available for macros. As a consequence, it's almost impossible to redefine the `\input` primitive without breaking its syntax (one would have to parse the characters one by one, and I'm not ready to do so...). ***FiNK*** currently does not follow auxiliary files either.

5 Changes

v2.0 New macros `\finkdir`, `\finkbase`, `\finkext` and `\finkpath` suggested by Alain Schremmer
New options `mainext` and `maindir`, use `kvoptions` for options management

v1.2 Fixed conflict with `\includegraphics`, reported by Jim Crumley

v1.1 Fixed missing 3rd arg to `\PackageError` call from `\finkextension`

6 The Code

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fink}[2007/06/11 2.0
                       Keep track of the current filename]

\RequirePackage{kvoptions}
\SetupKeyvalOptions{family=fnk,prefix=fnk@}

```

6.1 Main file initial settings

maindir
mainext

```
\DeclareStringOption[\@currdir]{maindir}
\DeclareStringOption[tex]{mainext}

```

The following is for backward compatibility only (not documented anymore). It provides support for the old `tex` and `ltx` options (still functionnal), and for the `\finkextension` macro. However, this macro is now defined to trigger an error, begging the user to use the new option instead.

```
\newcommand*\@fink@mainext[1]{\setkeys{fnk}{mainext={#1}}}
\newcommand*\fink@mainext{%
  \expandafter\@fink@mainext\expandafter{\CurrentOption}}
\DeclareVoidOption{tex}{\fink@mainext}
\DeclareVoidOption{ltx}{\fink@mainext}

\newcommand*\finkextension[1]{%
  \PackageError{FiNK}{%
    \protect\finkextension\space shouldn't be used anymore.\MessageBreak
    Please use the 'mainext' package option instead.}{%
    No big deal right ?\MessageBreak
    Type X to quit and modify your source.}}
\@onlypreamble\finkextension

\ProcessKeyvalOptions*

```

6.2 File's name components macros

`\finkdir` `\finkbase` `\finkext` `\finkfile` `\finkpath`

We declare the user-level macros here. `\fink@file` is used to compute file names, possibly with no extension.

```
\newcommand*\finkdir{\fnk@maindir}
\newcommand*\finkbase{\jobname}
\newcommand*\finkext{\fnk@mainext}

\newcommand*\finkfile{}
\newcommand*\fink@file[2]{#1\ifx\\#2\\\else.#2\fi}
\xdef\finkfile{\fink@file{\jobname}{\fnk@mainext}}

\newcommand*\finkpath{}
\xdef\finkpath{\finkdir\finkfile}

\PackageInfo{FiNK}{main file set to "\finkpath"}

```

6.3 Commands overriding

`\fink@prepare`

This macro prepares the name of next file to be input. We arrange to setup a complete filename, including directory and extension.

As of version 1.2, this macro performs in a group of its own. This fixes a problem that appeared when using `\includegraphics` with a filename with an explicit extension. `\includegraphics` calls `\filename@parse` itself, so it is important that the same call in `\fink@prepare` only have a local effect, just the time to compute the new values for the `\fink@next*` macros.

```
\newcommand*\fink@prepare[1]{%
  {\filename@parse{#1}%
    \xdef\fink@nextdir{%
```

```
      \ifx\filename@area\@empty%
        \fnk@maindir%
      \else%
        \fnk@maindir\filename@area%
      \fi}%
    \xdef\fink@nextbase{\filename@base}%
    \xdef\fink@nextext{\ifx\filename@ext\relax tex\else\filename@ext\fi}%
    \xdef\fink@nextfile{\fink@file{\fink@nextbase}{\fink@nextext}}%
    \xdef\fink@nextpath{\fink@nextdir\fink@nextfile}}}

```

`\fink@input` `\fink@restore` These macros are defined for a convenient use of `\expandafter`. They save and restore the current filename. Remember that `\@@input` is LaTeX's redefinition of the TeX input primitive.

```
\newcommand*\fink@input{%
  \xdef\finkdir{\fink@nextdir}%
  \xdef\finkbase{\fink@nextbase}%
  \xdef\finkext{\fink@nextext}%
  \xdef\finkfile{\fink@nextfile}%
  \xdef\finkpath{\fink@nextpath}%
  \@@input\@filef@und}
\newcommand*\fink@restore[1]{%
  {\filename@parse{#1}%
    \xdef\finkdir{\filename@area}%
    \xdef\finkbase{\filename@base}%
    \xdef\finkext{\filename@ext}%
    \xdef\finkfile{\fink@file{\finkbase}{\finkext}}
    \xdef\finkpath{\finkdir\finkfile}}}

```

Note: in earlier versions, we redefined `\IfFileExists` to prepare the name of the next file, but this is bad because it can be used outside of ***FiNK***'s scope. We also redefined `\@input`, but neither `\include` nor `\input` use it.

`\InputIfFileExists` LaTeX's `\input` and `\include` commands use `\InputIfFileExists`, so let's redefine it here:

```
\long\def\InputIfFileExists#1#2{%
  \IfFileExists{#1}{%
    #2\@addtofilelist{#1}%
    \fink@prepare{#1}%
    \expandafter\fink@input%
    \expandafter\fink@restore\expandafter{\finkpath}}}

```

Well, I think that's it. Enjoy using ***FiNK***!