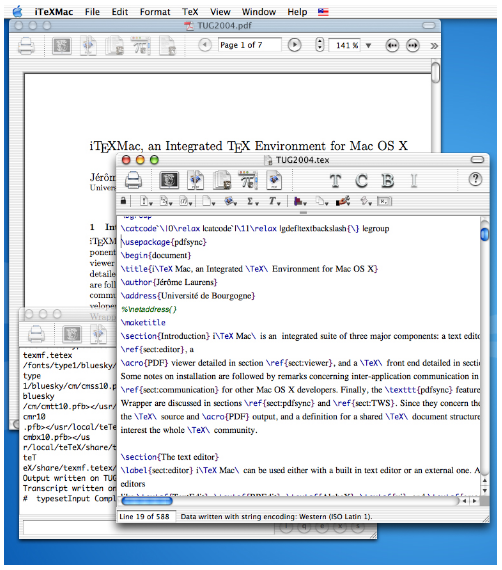
iTEXMac: An Integrated TEX Environment for Mac OS X

Jérôme Laurens
Université de Bourgogne
`jerome.laurens@u-bourgogne.fr`



```
\catcode`\|0\relax |catcode`|\11\relax |gdef|textbackslash{\} |egroup
\usepackage{pdfsync}
\begin{document}
\title{i\TeX Mac, an Integrated \TeX\ Environment for Mac OS X}
\author{Jérôme Laurens}
\address{Université de Bourgogne}
%\netaddress{}
\maketitle
\section{Introduction} i\TeX Mac\ is an  integrated suite of three major components: a text edit
\ref{sect:editor}, a
\acro{PDF} viewer detailed in section \ref{sect:viewer}, and a \TeX\ front end detailed in secti
Some notes on installation are followed by remarks concerning inter-application communication in
\ref{sect:communication} for other Mac OS X developers. Finally, the \texttt{pdfsync} feature
Wrapper are discussed in sections \ref{sect:pdfsync} and \ref{sect:TWS}. Since they concern the
the \TeX\ source and \acro{PDF} output, and a definition for a shared \TeX\ document structure
interest the whole \TeX\ community.

\section{The text editor}
\label{sect:editor} i\TeX Mac\ can be used either with a built in text editor or an external one. A
editors
```

```
Line 19 of 588    Data written with string encoding: Western (ISO Latin 1).
```

```
texmf.tetex
/fonts/type1/bluesky/
type
1/bluesky/cm/cmss10.p
bluesky
/cm/cmtt10.pfb></usr/
cmr10
.pfb></usr/local/teTe
cmbx10.pfb></us
r/local/teTeX/share/t
teT
eX/share/texmf.tetex/
Output written on TUG
Transcript written on
#  typesetInput Compl
```

1 Introduction

iTEXMac is an integrated suite of three major components: a text editor detailed in section 2, a PDF viewer detailed in section 3, and a TEX front end detailed in section 4. Some notes on installation are followed by remarks concerning inter-application communication in section 6 for other Mac OS X developers. Finally, the `pdfsync` feature and the TEX Wrapper are discussed in sections 7 and 8. Since they concern the synchronization between the TEX source and PDF output, and a definition for a shared TEX document structure, both will certainly interest the whole TEX community.

2 The Text Editor

iTEXMac can be used either with a built-in text editor or an external one. All standard text editors like TextEdit, BBEdit, AlphaX, vi, and emacs are supported and configuring iTEXMac for other editors is very easy, even when coming from the X11 world.

The built-in text editor comes with flavours similar to emacs and AlphaX modes. It relies on a plug-in architecture that allows very different kinds of user interfaces according to the type of the file being edited. Whereas AlphaX uses Tcl and emacs uses Lisp, iTEXMac utilizes the benefits of Objective-C bundles, giving plug-ins great potential power together with the application.

Among the standard features shared by advanced text editors (like key binding management, advanced regular expressions, command completion), an interesting feature of iTEXMac's text editor is the syntax parsing policy. The syntax highlighting deeply depends on the kind of text edited, whether it is Plain, LATEX or METAPOST (support for HTML is planned). The text properties used for highlighting include not only the color of the text, but also the font, the background color and some formatting properties.

Moreover, the command shortcuts that refer to mathematical and text symbols are replaced by the glyph they represent, thus replacing `\alpha` with α and so on. Conversely the built-in editor can show a character palette with 42 menus gathering text and mathematical symbols, as they would appear in the output. The editor thus serves as a graphical front-end to the standard LATEX packages, **`amsfonts.sty`**, **`amssymb.sty`**, **`mathbb.sty`**, **`mathrsfs.sty`**, **`marvosym.sty`** and **`wasysym.sty`**, which makes thousands of symbols available with just one click. The result is a text editor that contains much more WYSIWYG than others, with no source file format requirement.

There is also advanced management of string encoding, and iTEXMac supports more than 80 of them with an efficient user interface. The text files are scanned for hints about the text encoding:

LATEX	`\usepackage[`*encoding*`]{inputenc}`
ConTEXt	`\enableregime[`*encoding*`]`
emacs	`%-*-coding:`*character encoding*`;-*-`
	`%!iTeXMac(charset):` *character encoding*
	Mac OS X hidden internals

But this is not user friendly practice and will be enhanced by the forthcoming discussion of TEX wrappers in section 8.

Spell checking for TEX input is available with Rick Zaccone's LATEX aware Excalibur[1] and Anton

[1] `http://www.eg.bucknell.edu/~excalibr/`

Leuski's TEX aware cocoAspell[2], a port of the Free and Open Source spell checker aspell. The latter also knows about HTML code and is integrated to Mac OS X allowing iTEXMac to check spelling as you type, with misspelled words being underlined in red. While this service is provided to all applications for free, iTEXMac is the only one that truly enables the TEX support by managing the language and the list of known words on a file by file basis using TEX Wrappers.

3 The PDF Viewer

iTEXMac can be used either with a built-in PDF viewer or an external one. The built-in viewer lacks many advanced features of Acrobat or Preview, but it updates the display automatically when a PDF file has been changed externally. Moreover, it allows you to make selections and export them to other applications, like Word, TextEdit or Keynote for example. Finally, it has support for the useful PDF synchronization discussed below and is very well integrated in the suite.

iTEXMac can open PS, EPS and DVI files with a double click, by first converting them to PDF. It thus plays the role of a PostScript or a DVI viewer. This feature is now partially obsolete since Mac OS X version 10.3 provides its own PS to PDF translator used by the Preview application shipped with the system.

4 The TEX Front End

This component of the software serves two different purposes. On one hand it is a bridge between the user and the utilities of a standard TEX distribution: a graphical user interface for the commands `tex`, `latex`, `pdftex`, and so on. On the other hand, it has to properly manage the different kinds of documents one wants to typeset.

Actually, the iTEXMac interface with its underlying TEX distribution is fairly simple. Five basic actions are connected to menu items, toolbar buttons or command shortcuts to

- typeset (e.g., running `latex` once, or twice in advanced mode)
- make the bibliography (e.g. running `bibtex`)
- make the index (e.g., running `makeindex`)
- render graphics (e.g., running `dvipdf`)

All these actions are connected to shell scripts stored on a per document basis. If necessary, the user can customize them or even change the whole process by inserting an in-line instruction at the very beginning of a source file. For example, the following directive, if present, will run `pdflatex` in escape mode.

```
%!iTeXMac(typeset):  pdflatex
 --shell-escape $iTMInput
```

The `makeindex` and `bibtex` command options can be set from panels, and other commands are supported. Moreover, the various log files are parsed, warnings and errors are highlighted with different colors and HTML links point to lines where an error occurred. Some navigation facilities from log file to output are also provided, a string like `[`*a number...*`]`, pointing to the output page.

As for documents, iTEXMac manages a list of default settings that fit a wide range of situations, including for example

- LATEX documents with DVI, PS or PDF engines
- books
- METAPOST documents
- ConTEXt documents
- HTML documents
- B. Gaulle's French Pro documents.

Users can extend this list with a built-in editor, adding support for MusicTEX, maybe `gcc`, and so on.

5 Installing TEX and iTEXMac

The starting point for a detailed documentation is the MacOS X TEX/LATEX Web Site[3] where one will find an overview of the TEX related tools available on Mac OS X. As a graphical front-end, iTEXMac needs a TEX distribution to be fully functional. Gerben Wierda maintains on his site[4] the TEX Live[5] distribution and a set of useful packages. Other teTEX 2.0.2 ports are available from fink[6] (and through one of its graphical user interfaces, such as finkcommander[7]) and from Darwin Ports[8], through a CVS interface.

The official web site of iTEXMac is hosted by the Open Source software development SourceForge website at:

`http://itexmac.sourceforge.net/`

One can find in the download section the disk images for the following products:

- iTEXMac, both stable and developer release

[2] `http://cocoAspell.leuski.net/`

[3] `http://www.esm.psu.edu/mac-tex/`
[4] `http://www.rna.nl/tex.html`
[5] `http://www.tug.org/texlive/`
[6] `http://fink.sourceforge.net/pdb/package.php/tetex`
[7] `http://finkcommander.sourceforge.net/`
[8] `http://darwinports.opendarwin.org/ports/?by=cat&substr=print`

- an external editor for iTeXMac key binding
- the Hypertext Help With LaTeX wrapped as a searchable Mac OS X help file.
- the TeX Catalog On line wrapped as a searchable Mac OS X help file.
- the French LaTeX FAQ wrapped as a searchable Mac OS X help file.

An updater allows you to check easily for new versions. To install iTeXMac, just download the latest disk image archive, double click and follow the instructions in the read-me file.

Due to its Unix core, Mac OS X is no longer focused on only one user. To support multiple users, iTeXMac configuration files can be placed in different locations to change defaults for all, or just certain users. The search path is:

- the built-in domain as shipped with the application (with default, read-only settings)
- the network domain (`/Network/Library/Application Support/iTeXMac`), where an administrator can put material to override or augment the default behaviour of all the machines on a network
- the local domain (`/Library/Application Support/iTeXMac`), where an administrator can put material to override or augment the network or default behaviour
- the user domain (`~/Library/Application Support/iTeXMac`), where the user can put material to override or augment the local or default behaviour

This is a way to apply modifications to iTeXMac as a whole.

6 Inter-application Communication

This section describes how iTeXMac communicates with other components, in the hope that this syntax will also be used by other applications when relevant, to avoid the current situation where there are as many AppleScript syntaxes as there are available applications for TeX on the Macintosh. It also shows why iTeXMac integrates so well with other editors or viewers.

6.1 Shell Commands

iTeXMac acts as a server, such that other applications can send it messages. Each time it starts, iTeXMac installs an alias to its own binary code in `~/Library/TeX/bin/iTeXMac`. With the following syntax,[9] either from the command line, an AppleScript or shell script, one can edit a text file at the location corresponding to the given line and column numbers

```
~/Library/TeX/bin/iTeXMac edit -file
  "filename"
    -line lineNumber -column colNumber
```

The following syntax is used to display a PDF file at the location corresponding to the given line column and source file name

```
~/Library/TeX/bin/iTeXMac display -file
  "filename.pdf"
    -source "sourcename.tex"
    -line lineNumber -column colNumber
```

6.2 AppleScript

The same feature is implemented using this scripting language. It would be great for the user if other TeX front ends on Mac OS X would implement the same syntax.

```
tell application "iTeXMac" to edit
  "filename.tex"
    at line lineNumber column colNumber

tell application "iTeXMac" to display
  "filename.pdf"
    at line lineNumber column colNumber
    in source "Posix source name.tex"
```

iTeXMac support for AppleScript likewise covers the compile, bibliography and index actions. They are not given here since there is no Apple Events Suite dedicated to TeX. However, configuration files and instructions are given to let third-party applications like Alpha X, BBEdit or emacs control iTeXMac using those scripts.

6.3 HTML

iTeXMac implements support for a URL scheme named `file-special` for editing, updating or displaying files, for example

```
file-special://localhost/"filename.tex";
    action=edit;line=lineNumber;
    column=columnNumber

file-special://localhost/"filename.pdf";
    action=display;line=lineNumber;
    column=columnNumber;
        source="Posix source name.tex"
```

will ask iTeXMac to edit a TeX source file or display the given file (assumed to be PDF) and when synchronization information is available, scroll to the location corresponding to the given line and column in source (assumed to be TeX). This allows adding dynamic links in HTML pages, in a TeX tutorial for example.

[9] These commands should be entered all on one line. They are broken here due to the narrow *TUGboat* columns.

7 The pdfsync Feature

7.1 About Synchronization

As the TEX typesetting system heavily relies on a page description language, there is no straightforward correspondence between a part of the output and the original description code in the input. A workaround was introduced a long time ago by commercial TEX frontends Visual TEX[10] and TEXtures[11] with a very efficient implementation. Then LATEX users could access the same features — though in a less-efficient implementation — through the use of `srcltx.sty`, which added source specials in the DVI files. The command line option `-src-specials` now gives this task to the TEX typesetting engine.

When used with an external DVI viewer or an external text editor, through an X11 server or not, iTEXMac fully supports this kind of synchronization feature.

For the PDF file format, Piero d'Ancona and the author elaborated a strategy that works rather well for Plain TEX, ConTEXt and LATEX users. While typesetting a `foo.tex` file with LATEX for example, the `pdfsync` package writes extra geometry information in an auxiliary file named `foo.pdfsync`, subsequently used by the front ends to link line numbers in source documents with locations in pages of output PDF documents. iTEXMac and TEXShop[12] both support `pdfsync`.

The official `pdfsync` web site is:

`http://itexmac.sourceforge.net/pdfsync.html`

It was more convenient to use an auxiliary file than to embed the geometric information in the PDF output using `pdftex` primitives. The output file is not polluted with extraneous information and the front ends need not parse the PDF output to retrieve such metrics.

7.2 The pdfsync Mechanism

A macro is defined to put `pdfsync` anchors at specific locations (for hbox's, paragraphs and maths). There are essentially three problems we must solve: the position of an object in the PDF page is not known until the whole page is composed, the objects don't appear linearly in the output[13] and finally, an input file can be entirely parsed long before its contents are shipped out. To solve these, at each `pdfsync` anchor the known information (line number and source file name) is immediately written to the `pdfsync` auxiliary file and the unknown information (location and page number) will be written only at the next ship out.

[10] `http://www.micropress-inc.com/`
[11] `http://www.bluesky.com/`
[12] `http://www.uoregon.edu/~koch/texshop`
[13] The footnotes objects provide a good example.

7.3 The .pdfsync File Specifications

This is an ASCII text file organized into lines. There is no required end of line marker format from among the standard ones used by operating systems.

Only the two first lines described in table 1 are required, the other ones are optional. The remaining lines are described according to their starting characters, they consist of 2 interlaced streams. A synchronous one detailed in table 2 is obtained with `\immediate\write`s and concerns the input information. An asynchronous one detailed in table 3 is obtained with delayed `\write`s and concerns the output information.

The correspondence between the two kinds of information is made through a record counter, which establishes a many-to-many mapping from line numbers in TEX sources to positions in PDF output.

7.4 Known Problems

Unfortunately, the various `pdfsync` files for Plain, LATEX or ConTEXt are not completely safe. Some compatibility problems with existing macro packages may occur. Moreover, sometimes `pdfsync` actually influences the final layout; in a case like that, it should only be used in the document preparation stage.

Another mechanism widely used by ConTEXt makes `pdfsync` sometimes inefficient, where the macro expansion only occurs long after it has been parsed, such that the `\inputlineno` is no longer relevant and the significant line number is no longer accessible. This makes a second argument for the implementation of the `pdfsync` feature at a very low level, most certainly inside the `pdftex` engine itself.

8 TWS: A TEX Wrapper Structure

In general, working with TEX seems difficult due to the numerous auxiliary files created. Moreover, sharing TEX documents is often delicate as soon as we do not use very standard LATEX. The purpose of this section is to lay the foundation for the TEX Wrapper Structure, which aims to help the user solve these problems.

First, it is very natural to gather all the files related to one TEX document in one folder we call a TEX Wrapper. The file extension for this directory is `texd`, in reference to the `rtf` and `rtfd` file extensions already existing on Mac OS X. The contents of a TEX wrapper named *document*`.texd` is divided according to different criteria:

Line	Format	Description	Comment
1st	*jobName*	*jobName*: case sensitive TEX file name	In general, the extensionless name of the file as the result of an `\immediate\write\file{\jobname}`
2nd	`version` *V*	*V*: a 0 based non-negative integer	The current version is 0

Table 1: `pdfsync` required lines format

Line	Format	Description	Comment
“b”	`b` *name*	*name*: TEX file name	TEX is about to begin parsing *name*, all subsequent line and column numbers will refer to *name*. The path is relative to the directory containing the `.pdfsync` file. Path separators are the Unix “`/`”. The file extension is not required, “tex” is the default if necessary. Case sensitive.
“e”	`e`		The end of the input file has been reached. Subsequent line and column numbers now refer to the calling file. Optional, but must match a corresponding “b” line.
“l”	`l` *R L* `l` *R L C*	*R*: record number, *L*: line number, *C*: optional column number.	

Table 2: `pdfsync` line specifications of the synchronous stream

Line	Format	Description	Comment
“s”	`s` *S*	*S*: physical page number	TEX is going to ship out a new page.
“p” “p*”	`p` *R x y* `p*` *R x y*	*R*: record number, *x*: horizontal coordinate, *y*: vertical coordinate.	Both coordinates are respectively given by `\the\pdflastxpos` and `\the\pdflastypos`

Table 3: `pdfsync` line specification of the asynchronous stream

- required data (source, graphics, bibliography database)
- helpful data and hints (tex, bibtex, makeindex options, known words)
- user-specific data
- front-end-specific data
- cached data
- temporary data

It seems convenient to gather all the non-required information in one folder named *document*`.texd/` *document*`.texp` such that silently removing this directory would cause no harm. As a consequence, no required data should stay inside *document*`.texp`, and this is the only rule concerning the required data. The `texp` file extension stands for “TEX Project”.

In tables 4 to 9 we show the core file structure of the *document*`.texp` directory. This is a minimal definition involving only string encoding and spelling information because there is no consensus yet among users and all the developers of TEX solutions, on Mac OS X at least. We make use of the XML property list data format storage as defined by `http://www.apple.com/DTDs/PropertyList-1.0.dtd`

Name	Contents
`Info.plist`	XML property list for any general purpose information wrapped in an `info` dictionary described in table 5. Optional.
spellingKey.`spelling`	XML property list for lists of known words wrapped in a `spelling` dictionary defined in table 9 and uniquely identified by *spellingKey*. This format is stronger than a simple comma separated list of words. Optional.
`frontends`	directory dedicated to front-ends only.
`frontends/`*name*	private directory dedicated to the front-end identified by *name*. The further contents definition is left under the front-end responsibility.
`users`	directory dedicated to users. Should not contain any front-end specific data.
`users/`*name*	directory dedicated to the user identified by *name* (not its login name). Not yet defined, but private and preferably encrypted.

Table 4: Contents of the TEX Project directory *document*.`texp`

Key	Class	Contents
`isa`	String	Required with value: `info`
`version`	Number	Not yet used but reserved
`files`	Dictionary	The paths of the files involved in the project wrapped in a `files` dictionary. Optional.
`properties`	Dictionary	Attributes of the above files wrapped in a `properties` dictionary. Optional.
`main`	String	The *fileKey* of the main file, if relevant, where *fileKey* is one of the keys of the `files` dictionary. Optional.

Table 5: `info` dictionary description.

Key	Class	Contents
fileKey	String	The path of the file identified by the string *fileKey*, relative to the directory containing the TEX project. No two different keys should correspond to the same path.

Table 6: `files` dictionary description.

However, this mechanism doesn't actually provide the concrete information needed to typeset properly (engine, format, output format). For that we can use `Makefiles` or shell scripts either embedded in the TEX Wrapper itself or shipped as a standard tool in a TEX distribution. This latter choice is less powerful but much more secure. Anyway, a set of default actions to be performed on a TEX Wrapper should be outlined (compose, view, clean, archive...).

Technically, iTEXMac uses a set of private, built-in shell scripts to typeset documents. If this is not suitable, customized ones are used instead, but no warning is given then. No security problem has been reported yet, most certainly because such documents are not shared.

Notice that iTEXMac declares `texd` as a document wrapper extension to Mac OS X, which means that *document*.`texd` folders are seen by other applications just like other single file documents, their contents is hidden at first glance. Using another file extension will prevent this Mac OS X feature without losing the benefit of the TEX Wrapper Structure.

A final remark concerns the version control system in standard use among TEX users. In the current definition, only one directory level should be

Key	Class	Contents
fileKey	Dictionary	Language, encoding, spelling information and other attributes wrapped in an `attributes` dictionary described in table 8. *fileKey* is one of the keys of the `files` dictionary.

Table 7: `properties` dictionary description.

Key	Class	Contents
`isa`	String	Required with value: `attributes`
`version`	Number	Not yet used but reserved
`language`	String	According to latest ISO 639. Optional.
`codeset`	String	According to ISO 3166 and the IANA Assigned Character Set Names. If absent the standard C++ locale library module is used to retrieve the `codeset` from the `language`. Optional.
`eol`	String	When non void and consistent, the string used as end of line marker. Optional.
`spelling`	String	One of the *spellingKeys* of table 4, meaning that *spellingKeys*.`spelling` contains the list of known words of the present file. Optional.

Table 8: `attributes` dictionary description

Key	Class	Contents
`isa`	String	Required with value: `spelling`
`version`	Number	Not yet used but reserved
`words`	Array	The array of known words

Table 9: `spelling` dictionary description.

supported in a *document*.`texp` folder. The contents of the `frontend` and `users` should not be monitored.

9 Nota Bene

Some features discussed here are still in the development stage and are still being tested and validated (for example, advanced syntax highlighting and full TWS support).