A beginner's guide to file encoding and TEXShop

Herbert Schulz and Richard Koch

Abstract

A common problem TEX users face when opening and typesetting files is that the text displayed either in the source or in the typeset document or both is not what should be there; characters are scrambled and improper characters appear. This is usually an *encoding* problem — either the editor or TEX or both do not interpret the input correctly.

This document is meant as a first introduction to file encodings. It is definitely *not* meant as an exhaustive document, and deals only with the most common encodings in use today.

While the following document was originally written for distribution with the TEXShop editor (a.k.a. front end) running on Mac systems, other front ends use a similar directive, and the general discussion about file encodings is valid no matter what editor you use.

1 What is a file encoding?

While we usually think of the .tex source file as containing characters, in reality this source, like all computer files, is just a long stream of whole numbers, each (nowadays) from 0 through 255. Computer scientists call these whole numbers *bytes*.

All other computer data must be encoded in one way or another into bytes. The most common encoding of ordinary text into bytes is called ASCII; it encodes most of the characters found on an ordinary American typewriter. For instance, the characters 'A' through 'Z' are encoded as 65 through 90, the characters 'a' through 'z' become 98 through 123. The space character is encoded as byte 32, and numerals, parentheses, and punctuation characters encode as other bytes.

Originally, TEX required ASCII input. While this was sufficient in the United States, it proved cumbersome in Western Europe, where accents, umlauts, upside down question marks, and the like are common; macros were needed to construct those characters and that broke hyphenation. More difficult problems arose when TEX was used in the Near and Far East.

The ASCII encoding only uses bytes from 0 through 127. Thus the door was open to encode other characters using bytes 128 through 255. Many different single-byte encodings now exist to display additional characters using these bytes.

2 Extending the character table

The three most often used extended single-byte encodings on the Mac are MacOSRoman, IsoLatin1 and IsoLatin9.[1]

The MacOSRoman encoding is left over from the days before OS X and, as expected, exclusive to Mac computers. Its use is no longer encouraged.

The IsoLatin1 encoding extends the ASCII encoding with the accented characters used in Western European languages.

IsoLatin9 primarily adds the Euro symbol, €, to the IsoLatin1 encoding along with a few other changes.

2.1 Other encodings used with TEX

Other fairly common encodings include IsoLatin2 for central European languages, IsoLatin5 for Turkish and IsoLatinGreek (also called Iso8859-7) for Greek. Several different encodings are available for Russian and other languages using Cyrillic. Additional encodings are available for Korean and Chinese, but Far Eastern languages use thousands of symbols, so these encodings are not very satisfactory.

2.2 Windows stuff

Windows Latin 1 is a version of IsoLatin1 with some characters in different code locations as defined by Microsoft. Thus, folks running Windows can end up with files in this encoding.

2.3 A crucial flaw

The various encodings were developed independently by computer companies as their products were sold in more and more countries.

Unfortunately (but unavoidably), text files do not have a header specifying the encoding used by the file. Thus there is no way for TEXShop to automatically adjust the encoding as various files are input. Some text editors have built-in heuristics to try to guess the correct encoding, but TEXShop does not use these heuristics because they work only 90% of the time and an incorrect guess can lead to havoc.

3 Unicode

As the computer market expanded across the world, computer companies came to their senses and created a consortium to develop an all-encompassing standard, called Unicode. The goal of Unicode is to encode all symbols commonly used across the world, including Roman, Greek, Cyrillic, Arabic, Hebrew, Chinese, Japanese, Korean, and many others. Unicode even has support for Egyptian hieroglyphics and

[1] We will use the same notation as for the TEXShop encoding directive in this document. See the table on page 176.

relatively recently added support for mathematical symbols.

All modern computer systems, including Macintosh, Windows, GNU/Linux and other Unix, now support Unicode. Internally, TEXShop and many other editors represent characters using Unicode and thus can accept text that is a combination of Roman, Greek, Cyrillic, Arabic, Chinese, and other languages. TEXShop even understands that Arabic, Hebrew, and Persian are written from right to left. To input these extra languages, activate additional keyboards using the System Preferences Keyboard Pane. This Pane changed in recent versions of OS X; in El Capitan, select a keyboard on the left, or click '+' below the list to see a list of additional languages and add their keyboards.

3.1 Unicode representations

Because it has far more than 256 symbols, Unicode defines symbols using much larger integers, using more than one byte. Unicode defines the "internal" structure of these numbers, but gives several different ways to represent the numbers on computers. By far the most popular Unicode encoding nowadays is UTF-8, which uses a sequence of 8-bit bytes, but UTF-16 (using a sequence of 16-bit chunks) and others are also available. (All the Unicode representations are equivalent; the multiple representations exist for historical and other reasons beyond this short note.)

The great advantage of UTF-8 is that ordinary ASCII characters retain their single-byte form in the encoded file. Consequently, ordinary ASCII files remain valid as UTF-8 files. With most single-byte encodings like IsoLatin1, IsoLatin9, etc., any sequence of bytes forms a legal file. If you open such a file with the wrong encoding, the file will be read, but some of the symbols will be wrong. For example, if someone in Germany using IsoLatin9 collaborates with someone in the U.S. using MacOSRoman, and their paper is written in English, they may not notice the mismatch until they proofread the references and discover that accents and umlauts have gone missing.

However, not all sequences of bytes form valid UTF-8 files, because non-ASCII symbols are converted into bytes using a somewhat complicated code. In the previous example, if the German collaborator uses IsoLatin9 and includes non-ASCII characters, such as those with umlauts, in the document and the American collaborator uses UTF-8, TEXShop will report an error when it tries to open the IsoLatin9 file in UTF-8. TEXShop will then display an error message and offer to open the file in a "fallback" single-byte encoding, currently IsoLatin9 (not configurable).

On the other hand, both authors of this document use UTF-8 Unicode as our default encoding, turning that message to our advantage. UTF-8 preserves everything typed in TEXShop, so there are no puzzling character losses. HTML and other code is usually saved in UTF-8, so TEXShop can be used as a more general text editor. Moreover, if a TEX file from an external source is not in UTF-8, we get the warning above. The trick is then to let TEXShop open the file in the "fallback" encoding, IsoLatin9, and examine the file for an inputenc line which tells you what encoding was actually used. Then close the file *without making any changes* and re-open it using the Open dialog and manually choose the correct encoding. Once the file is open with the correct encoding you may add the TEXShop encoding directive line for that encoding and save it for future use.

Using UTF-8 Unicode has become so advantageous that TEXShop 4.00 and later use this encoding as the default, out of the box,[2] encoding.

3.2 Encoding vs. formatting

All of the encoding methods discussed here, including Unicode, are irrelevant to italics, underlining, font size, font color, etc. They just define characters as numbers. It is up to users to specify additional attributes in some other way. For example, when Apple's TextEdit program is used in *Plain Text* mode, a user can change the font or font size for an entire document, but not for individual sections of the document. If the document is saved to disk and then reloaded, the font changes are lost. On the other hand, a word processor like Microsoft Word or Apple Pages has much more control over fonts, font size and the like. These programs output text in a proprietary format readable only by that program, but the file does preserve the extra attribute information.

While all modern computers support Unicode, particular fonts (nearly always) have symbols for only a small portion of the Unicode world. Fonts should have a special character, often a box, to indicate that a character is missing. Thus if you want to write in, say, Arabic or Hebrew, you must choose a font which contains these symbols. Modern computers support a great range of symbols because the computer business covers the world, but it may still be hard to find a font covering obscure Unicode symbols.

[2] If you switch to the latest TEXShop version and have already reset the default encoding in TeXShop → Preferences, your selection will be maintained.

4 Two sides of the story: TeXShop and TeX

Once a user selects an appropriate encoding, the user must configure both TeXShop and the appropriate TeX engine to use that encoding. Different sets of problems arise with these two tasks.

Users in the United States and other English speaking countries can often ignore encodings altogether. The default TeXShop encoding supports ASCII, and TeX and LaTeX have supported ASCII from the beginning. So there is nothing to do.

Users in Western Europe must take slightly more care. The current default TeXShop encoding, UTF-8 Unicode, will be sufficient for their needs. But they must configure TeX and LaTeX as described below, and carefully choose fonts which support the needed accents, umlauts, and the like. The required steps are easy.

Users in Russia and Eastern Europe must take similar steps, but the authors of this paper are not knowledgeable about correct configurations, so we suggest getting help from friends already using TeX.

Users in the Far East and Middle East, and scholars working with multi-language projects, will need to consult other sources for detailed configurations. These users should certainly examine XeTeX and LuaTeX, because these extensions of TeX use Unicode directly and are much more capable of handling languages where Unicode becomes essential. Both XeTeX and LuaTeX can typeset almost all standard TeX and LaTeX source files, but have additional code for Unicode support. One big problem with these languages is that appropriate fonts must be chosen which support the languages. To simplify that problem, both XeTeX and LuaTeX allow users to use the ordinary system fonts supplied with their computer.

5 Telling TeXShop what encoding to use to Load and Save source files

To set the default TeXShop encoding, open TeXShop Preferences. Select the Source tab. In the second column, find the Encoding section. This section contains a pull down menu; select the desired encoding from this menu. Select Western (ISO Latin 9) to get the IsoLatin9 encoding, useful in English speaking countries and Western Europe. You must select Unicode (UTF-8), the current default, or Unicode (UTF-16) if you want to preserve everything you can type into the TeXShop editor. If you pick any other encoding, there may be characters you can type in TeXShop which will be lost if you Save and then re-Load. On the other hand, UTF-8 may not work well with certain LaTeX packages, as explained later.

TeXShop has a mechanism to set the encoding of a particular file independent of the user's default choice, or of choices in the Load and Save panels. To set the encoding used to read or write a particular file to UTF-8, add the following line to the first twenty lines of the top of the file:

```
% !TEX encoding = UTF-8 Unicode
```

The easy way to do this is to select the Macro command Encoding. A dialog will appear from which the desired encoding can be selected, and after the dialog is closed, the line will be placed at the top of the file, replacing any existing encoding line.

If such a line exists, the indicated encoding will be used, overriding all other methods of setting the encoding, *unless* the option key is held down during the entire load or save operation.

Many users in Western Europe prefer to set IsoLatin9 as their default encoding so they can easily read files from collaborators, but include the line setting encoding to UTF-8 in file templates used to create files, so that their own files are encoded in UTF-8.

It is also possible to set the encoding used to read a file by Opening the file explicitly from within TeXShop. The resulting dialog has a pull-down menu at the bottom to select the encoding to be used for that particular file.[3] (Note that the "`% !TEX encoding =`" line overrides this command.)

Explicitly Saving a file from within TeXShop produces a Save Dialog with a similar pulldown menu to set the encoding.

Note: you *can't* easily change the encoding of a file. The best thing to do is copy the whole document into a new one and save that with the correct encoding. Using the TeXShop directive before saving the new file the first time is definitely recommended.

6 Telling LaTeX about file encodings

Your typesetting engine needs to know the encoding used to save each source file so the input source and the output glyphs are synchronized. For ordinary LaTeX, this is usually done by including a command like the following in the header of the source:

```
\usepackage[latin9]{inputenc}
```

Some values for other common encodings are given in the short table following.

This line is not needed when the source encoding is ordinary ASCII.

One valid value for encoding with `inputenc` is utf8. This line works in Western Europe, but not in situations requiring wider use of Unicode (because

[3] Under El Capitan you must first press the Options button to get to the pulldown menu.

TEXShop Open/Save dialogs	TEXShop encoding directive	LATEX `inputenc`
Unicode (UTF-8)	`UTF-8 Unicode`	`utf8`
Western (Mac OS Roman)	`MacOSRoman`	`applemac`
Western (ISO Latin 1)	`IsoLatin`	`latin1`
Central European (ISO Latin 2)	`IsoLatin2`	`latin2`
Turkish (ISO Latin 5)	`IsoLatin5`	`latin5`
Western (ISO Latin 9)	`IsoLatin9`	`latin9`
Mac Central European Roman	`Mac Central European Roman`	`macee`
Western (Windows Latin 1)	`Windows Latin 1`	`ansinew` or `cp1252`

Table 1: Partial encoding list of names in three contexts

the characters are lacking from TEX's usual fonts). When in doubt, it is useful to read the inputenc documentation. To do that, go to the TEXShop Help menu, select Show Help for Package, and fill in the requested Package with inputenc.

Users in Western Europe usually use *four* "related" commands in the header. Here are these four lines for users in Germany.

```
\usepackage[german]{babel}
\usepackage{lmodern}
\usepackage[T1]{fontenc}
\usepackage[latin9]{inputenc}
```

The first of these lines asks LATEX to use German conventions for dates, hyphenation, etc.

The second line tells LATEX to use the Latin Modern fonts. These fonts agree with Donald Knuth's Computer Modern fonts in the first 128 spots, but include additional accents, umlauts, upside down question marks, and so forth used in Western Europe.

The third line tells LATEX the connection between the input characters in the file and the glyphs in the fonts (i.e., the physical representation of the printed characters in the final document).

As explained above, the final line tells LATEX which encoding was used for the source file.

Users interested in more details should consult the documentation for babel, lmodern, and fontenc using TEXShop's Show Help for Package item in the Help Menu. The documentation is interesting, going into considerable historical detail about the evolution of font design in TEX.

7 Encodings understood by TEXShop

Table 1 shows the corresponding entries for some popular file/input encodings used with LATEX in TEXShop.

The 'Open/Save Dialogs' column shows the designation for the encodings in TEXShop's Open/Save Dialogs; you may have to click on the Options button to display the popup menu for encodings.

The 'Directive' column gives the designation used in TEXShop's encoding directive,

```
% !TEX encoding = xxxxx
```

where `xxxxx` is the designator you wish to use. If this line is in place before you first Save your source file, TEXShop will automatically save the file with the designated encoding. TEXShop will also automatically Open the file with that encoding when double clicked. We suggest you create a Template which contains the directive and use that to create new documents.

The '`inputenc`' column gives the optional argument for the LATEX `inputenc` package. As with the Directive, we suggest creating a Template which has the proper `inputenc` line for the corresponding encoding in the directive.

Good luck!

◇ Herbert Schulz and Richard Koch
tug.org/mactex