A Beginner's Guide
to
File Encoding & TeXShop
v0.3.0–2015/12/27

H. Schulz & R. Koch

1 Introduction

A common problem TeXShop users face when opening and typesetting files is that the text displayed in the Source or in the Typeset Document does not agree with what should be there; characters are scrambled and improper characters appear. This is usually an *encoding* problem — the Editor or TeX or both do not interpret the input correctly. This document is meant as a first introduction to encoding. It is definitely *not* meant as an exhaustive document, and deals only with the most common encodings in use today.

2 What is 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 between 0 and 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 all 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 used; 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 between 0 and 127. Thus the door was open to encode other characters using bytes 128 through 255. Many different encodings now exist to display additional characters using these bytes.

3 Extending the Character Table

The three most used extended 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 the Mac Computer. Its use is no longer encouraged.

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

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

---

[1]We will use the notation used for the TeXShop encoding directive in this document. See the table in section (8) on page 5.

3.1 Other Encodings Used with TₑX

Additional encodings include IsoLatin2 for Central European Languages, IsoLatin5 for Turkish and Iso8859-7 for Greek. Several different encodings are available for Russians and others using Cyrillic. Additional encodings are available for Korean and Chinese, but Far Eastern languages use far more than 256 symbols, so these encodings are not very satisfactory.

3.2 Windows Stuff

Windows Latin 1 is a version of IsoLatin1 with some characters in different code locations as defined by Microsoft Corp. You can run into this encoding when you get files from folks running Windows.

3.3 A Crucial Flaw

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

Unfortunately, text files do not have a header listing the encoding used to generate 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.

4 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 recently added support for Mathematical Symbols.

All modern computer systems, including the Macintosh, Windows, Linux and Unix, now support Unicode. Internally, TₑXShop and other Macintosh editors describe characters using Unicode and 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.

Because there are far more than 256 symbols, Unicode describes symbols using much longer integers. Unicode proscribes the "internal" structure of these numbers, but defines several different ways to write the text to disk. The most popular Unicode encoding is UTF-8, but UTF-16 and others are also available.

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 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 appear as usual, but some of the symbols will be wrong. 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 legal UTF-8 files, because non-ASCII symbols are converted into bytes using a somewhat complicated code. In the previous example, if German collaborator uses IsoLatin1 and the American collaborator uses UTF-8 and the German collabo-

rator includes non-ASCII like umlauts in the references, then 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 default encoding, currently IsoLatin9. New users find those error messages much more confusing than occasional incorrect symbols, and that is one reason that the default TEXShop encoding is not currently UTF-8.

On the other hand both of the authors have set UTF-8 Unicode as our default encoding. This encoding 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 a warning. The trick is then to let TEXShop open the file in 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 open it using the Open dialog and manually choosing 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.

All of the encoding methods discussed here, including Unicode, ignore italics, underlining, font size, font, color, etc. They just encode characters. 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's Pages, has much more control over fonts, font size and the like. These progams output text with a proprietary coding only readable by that program. But the file preserves the extra attribute information.

While all modern computers support Unicode, their font sets have symbols for only a small portion of the Unicode world. Many fonts have a special character, often a box, to indicate that a character is missing. Thus if you want to write in 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 obscure Unicode symbols may not be covered by any single provided font.

5 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, IsoLatin9, will be sufficient for their needs. But they must configure TEX and LATEX as described below, and carefully choose fonts which support 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 knowledgable 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 XƎTEX and LuaTEX, because these extensions of TEX use Unicode directly and are much more capable of handling languages where Unicode becomes essential. Both XƎTEX 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 X$_\exists$T$_E$X and LuaT$_E$X allow users to use the ordinary system fonts supplied with their computer.

6 Telling T$_E$XShop what encoding is used to Load and Save files.

To set the default T$_E$XShop encoding, open T$_E$XShop 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 ISO Latin 9 to get the current default encoding, useful in English speaking countries and Western Europe. You must select UTF-8 Uncode or UTF-16 Unicode if you want to preserve anything typed into the T$_E$XShop editor. If you pick any other encoding, you there will be characters you can type in T$_E$XShop which will be lost if you Save and then reLoad. On the other hand, UTF-8 does not work well with certain L^{A}T$_E$X packages, as explained later.

T$_E$XShop 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 an appropriate 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 T$_E$XShop. The resulting open dialog has a pulldown menu at the bottom selecting the encoding to be used for that particular file.[2] (Note that the "% !TEX encoding =" line overrides this command.)

Explicitly Saving a file from within T$_E$XShop 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 T$_E$XShop directive before saving the new file the first time is definitely recommended.

7 Telling L^{A}T$_E$X about File Encodings

Your typsetting 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 one in the header of the source:

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

Typical values for other encodings are given in the short table at the end of this document.

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

One of the legal values for encoding with inputenc is utf8. This line works in Western Europe, but not in situations requiring deep use of Unicode. When in doubt, it is useful to read the inputenc documentation. To do that, go to the T$_E$XShop Help menu, select Show Help for Package, and fill in the requested Package with inputenc.

---

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

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, and the link.

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 characters in the file and actual glyphs (i.e., the physical representation of the characters in the final document).

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

Uses 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.

8 Encodings understood by TEXShop.

The table given below 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 inputenc package. As with the Directive, I suggest you create a Template which has the proper inputenc line for the corresponding encoding in the directive.

Table 1: *Partial Encoding List*

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