Creating Scholarly Multilingual Documents Using Unicode, OpenType, and XeTeX

David J. Perry

September 11, 2010

Version 1.6

This is a work in progress. Please send comments or corrections to: hospes02@scholarsfonts.net.

Abstract

This document is intended for people who are new to TeX and/or to XeTeX and who have a particular need to prepare multilingual, Unicode-based documents that take advantage of OpenType features.

- Sections 1–3 provide the information that new users need to understand what TeX is all about and that will help them decide whether they wish to learn TeX. These sections will also be useful to anyone new to TeX even if Unicode and OpenType support are not important.
- Sections 4 and 5 provide guidance on getting started with TeX, using the TeXworks development environment.
- Sections 6 and 7 plus the Appendices show how to take advantage of the advanced Unicode and OpenType features available in XeTeX.

There is much information available online about TeX. Some of it is overly complex for beginners; at the same time, it is hard to find guidance about using Unicode and OpenType with XeTeX. This article is by no means a complete guide to TeX, but after reading it users should be able to get a basic document working using the TeXworks environment and know how to access OpenType features. They can then use other resources to learn more about TeX in general; some suggestions for this are provided.

Those experienced with TeX but new to XeTeX, Unicode, and OpenType can consult Sections 6 and 7 plus the Appendix on fontspec, while skipping the other sections.

This article was prepared using TEXworks. Body text is set in Linux Libertine, an excellent open-source font by Philipp H. Poll, available from http://linuxlibertine.sourceforge.net/. Names of LATEX packages are in Linux Biolinum, a sans-serif companion to Linux Libertine, and and code samples are in DejaVu Sans Mono.

Contents

List of Figures

In this document, names of TEX add-on packages are printed in a sans-serif font, and code snippets (what you would actually type in your editor) are in `monospaced type`. Blue text indicates a clickable link to a location within the document, while green text is used for external URLs, which are also clickable.

Important Note:
This article assumes that you are familiar with Unicode and smart font technology such as OpenType and AAT. If you are not, see the reference in the first paragraph on page 5 for a source of information.

.

---

1 Why TEX?

Building on the foundation of Unicode, OpenType fonts provide an abundance of features that make it possible to create high-quality typography in many languages. In addition, some of these features are important to scholars in fields in as classics, medieval studies, biblical studies, and linguistics. If you need basic information about Unicode and OpenType, particularly regarding the role that OpenType can play in scholarship, see my book about document preparation for scholars. It provides pointers to many other sources in addition to the information provided in the book itself. Details about the book are available from http://scholarsfonts.net.

As of September 2010, Windows users who want to take advantage of these advanced OpenType features have limited options. (The situation with Mac OS X is much better.[1]) These options include the commercial desktop publishing programs Quark Express and Adobe InDesign and the open source typesetting language TEX and its extension XƎTEX. Scribus, the open source desktop publishing app, does not yet support OT features for advanced typography. Microsoft Word 2007 supports a limited number of OpenType features, and Word 2010 a few more, but not enough to make them useful for many scholarly needs.

Quark and InDesign provide very good support for OpenType features, but they are expensive even if one qualifies for academic pricing. Some people also prefer, for philosophical reasons, to use and support open source software. TEX is an open-source typesetting language designed specifically to produce high-quality documents; it is widely used in mathematics but is less well known in other academic fields. XƎTEX is a version of TEX that can take advantage of Unicode and OpenType features, providing an alternative to the commercial products, but installing and learning this software can be intimidating. This document reflects my own experiences as a relative newcomer to TEX and is designed to help others who wish to try it. Note that this is not a complete tutorial on TEX; there are already many such resources available, as discussed below. This document focuses on what beginners who require Unicode and OpenType support need to know. Because Unicode and OpenType are relative newcomers to the TEX world, this information is not so easy to come by.

2 The most important thing to understand

TEX and its extensions are a typesetting language, not a word processor such as many of us use daily. It does not employ a graphical interface that gives an on-screen representation of the final product. This means that instead of selecting a word and typing CTRL-B or clicking a “Boldface” button, the user must type in commands while editing the document. Then the source code is run through a processor that produces the formatted output; nowadays, this will probably

[1] Mac and Linux users should see the note on page 11

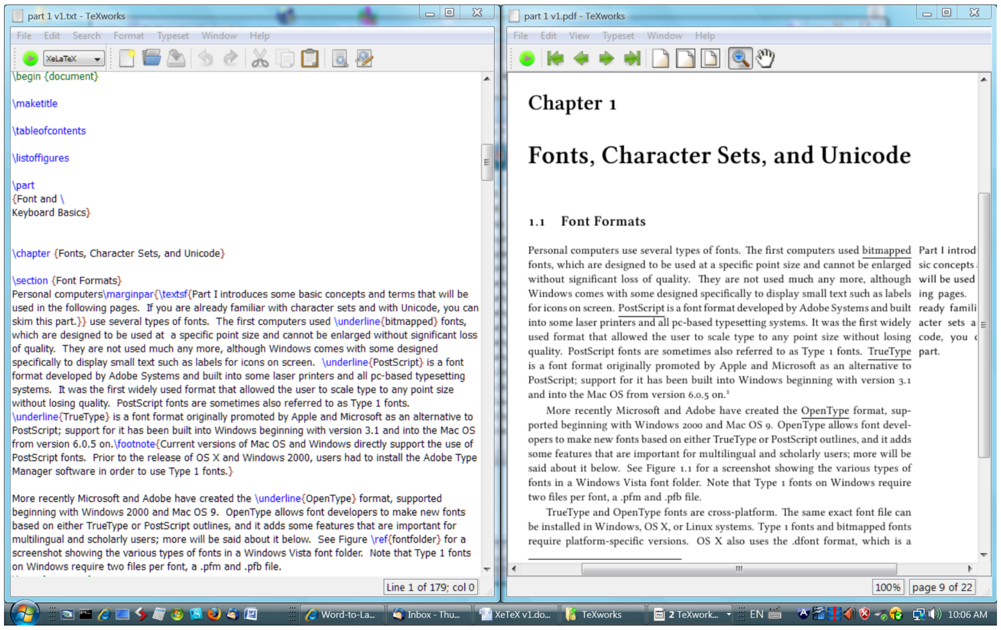
be a PDF file. Such systems were used on standalone typesetters prior to the personal computer era and were created for early PCs before Windows and Mac OS made development of WYSIWYG [2] programs such as InDesign possible.

There are some advantages to the TeX approach. It is designed to help users write well-structured documents by using tags such as \section and \subsection. It also separates, to a large extent, the content from the final design. TeX utilizes document classes, which are predefined layouts that specify such features as margins, headers and footers, font sizes, and so on. There are many of these available for different types of documents such as articles, books, etc. One can customize a document class or write a new one, but many users find that a document class prepared by a knowledgeable designer produces very good results, perhaps better than they could produce on their own using a word processor. Finally, it should be noted that while word processors have gotten more and more bloated, TeX is still small and efficient, even though it can do everything the word processors can. A MiKTeX download is about 83 megabytes, which includes many extra packages and their documentation; the actual files needed to compile a document are very much smaller. Compare this with the 500+ megabytes needed for a typical word processor. Because TeX source code is composed exclusively of plain text, these files are also very small, compared to those produced by most of today's word processors. A TeX installation and some source files can easily fit onto a flash drive. For more about the history of TeX and all the benefits of using it, see http://www.ctan.org/what_is_tex.html.

The screen shot in Figure 1 below shows a typical editing session using the TeXworks development environment, with the source code on the left and the output in PDF format displayed on the right.

Figure 2 (page 8) shows a larger view of a TeXworks editing window. The TeX typesetting commands are colored, and at the bottom of the window appears a panel (with green text) displaying the messages that the compiler generates when it processes the source code. These messages can very useful in tracking down errors in your source code. Some people are put off by the presence of the many tags that one must enter in TeX source code. This is very different from a WYSIWYG program and does take some getting used to. If the tags bother you and take the focus off your writing, I suggest using a familiar word processor to write and edit; don't waste time with fancy formatting since that will be done later in TeX. Once the text is finished, copy it and paste it into TeXworks (or whatever editor you use). This will remove any formatting you applied in the word processor. Or save the file as plain text and open it directly in TeXworks. You can then add the TeX tags and typeset the document. As you get more experienced, you'll probably find it's faster to add the tags as you go along, and perhaps you will find them less distracting. Despite the need to shift to a new paradigm, people who need sophisticated typography—particularly via OpenType features—should at least consider

[2]What You See Is What You Get



```
\begin {document}

\maketitle

\tableofcontents

\listoffigures

\part
{Font and \
Keyboard Basics}


\chapter {Fonts, Character Sets, and Unicode}

\section {Font Formats}
Personal computers\marginpar{\textsf{Part I introduces some basic concepts and terms that will be
used in the following pages.  If you are already familiar with character sets and with Unicode, you can
skim this part.}} use several types of fonts.  The first computers used \underline{bitmapped} fonts,
which are designed to be used at  a specific point size and cannot be enlarged without significant loss
of quality.  They are not used much any more, although Windows comes with some designed
specifically to display small text such as labels for icons on screen.  \underline{PostScript} is a font
format developed by Adobe Systems and built into some laser printers and all pc-based typesetting
systems.  It was the first widely used format that allowed the user to scale type to any point size
without losing quality.  PostScript fonts are sometimes also referred to as Type 1 fonts.
\underline{TrueType} is a font format originally promoted by Apple and Microsoft as an alternative to
PostScript; support for it has been built into Windows beginning with version 3.1 and into the Mac OS
from version 6.0.5 on.\footnote{Current versions of Mac OS and Windows directly support the use of
PostScript fonts.  Prior to the release of OS X and Windows 2000, users had to install the Adobe Type
Manager software in order to use Type 1 fonts.}

More recently Microsoft and Adobe have created the \underline{OpenType} format, supported
beginning with Windows 2000 and Mac OS 9.  OpenType allows font developers to make new fonts
based on either TrueType or PostScript outlines, and it adds some features that are important for
multilingual and scholarly users; more will be said about it below.  See Figure \ref{fontfolder} for a
screenshot showing the various types of fonts in a Windows Vista font folder.  Note that Type 1 fonts
on Windows require two files per font, a .pfm and .pfb file.
```

Chapter 1

Fonts, Character Sets, and Unicode

1.1 Font Formats

Personal computers use several types of fonts. The first computers used bitmapped fonts, which are designed to be used at a specific point size and cannot be enlarged without significant loss of quality. They are not used much any more, although Windows comes with some designed specifically to display small text such as labels for icons on screen. PostScript is a font format developed by Adobe Systems and built into some laser printers and all pc-based typesetting systems. It was the first widely used format that allowed the user to scale type to any point size without losing quality. PostScript fonts are sometimes also referred to as Type 1 fonts. TrueType is a font format originally promoted by Apple and Microsoft as an alternative to PostScript; support for it has been built into Windows beginning with version 3.1 and into the Mac OS from version 6.0.5 on.[1]

Part I introd sic concepts will be used ing pages. ready famili acter sets a code, you c part.

More recently Microsoft and Adobe have created the OpenType format, supported beginning with Windows 2000 and Mac OS 9. OpenType allows font developers to make new fonts based on either TrueType or PostScript outlines, and it adds some features that are important for multilingual and scholarly users; more will be said about it below. See Figure 1.1 for a screenshot showing the various types of fonts in a Windows Vista font folder. Note that Type 1 fonts on Windows require two files per font, a .pfm and .pfb file.

TrueType and OpenType fonts are cross-platform. The same exact font file can be installed in Windows, OS X, or Linux systems. Type 1 fonts and bitmapped fonts require platform-specific versions. OS X also uses the .dfont format, which is a

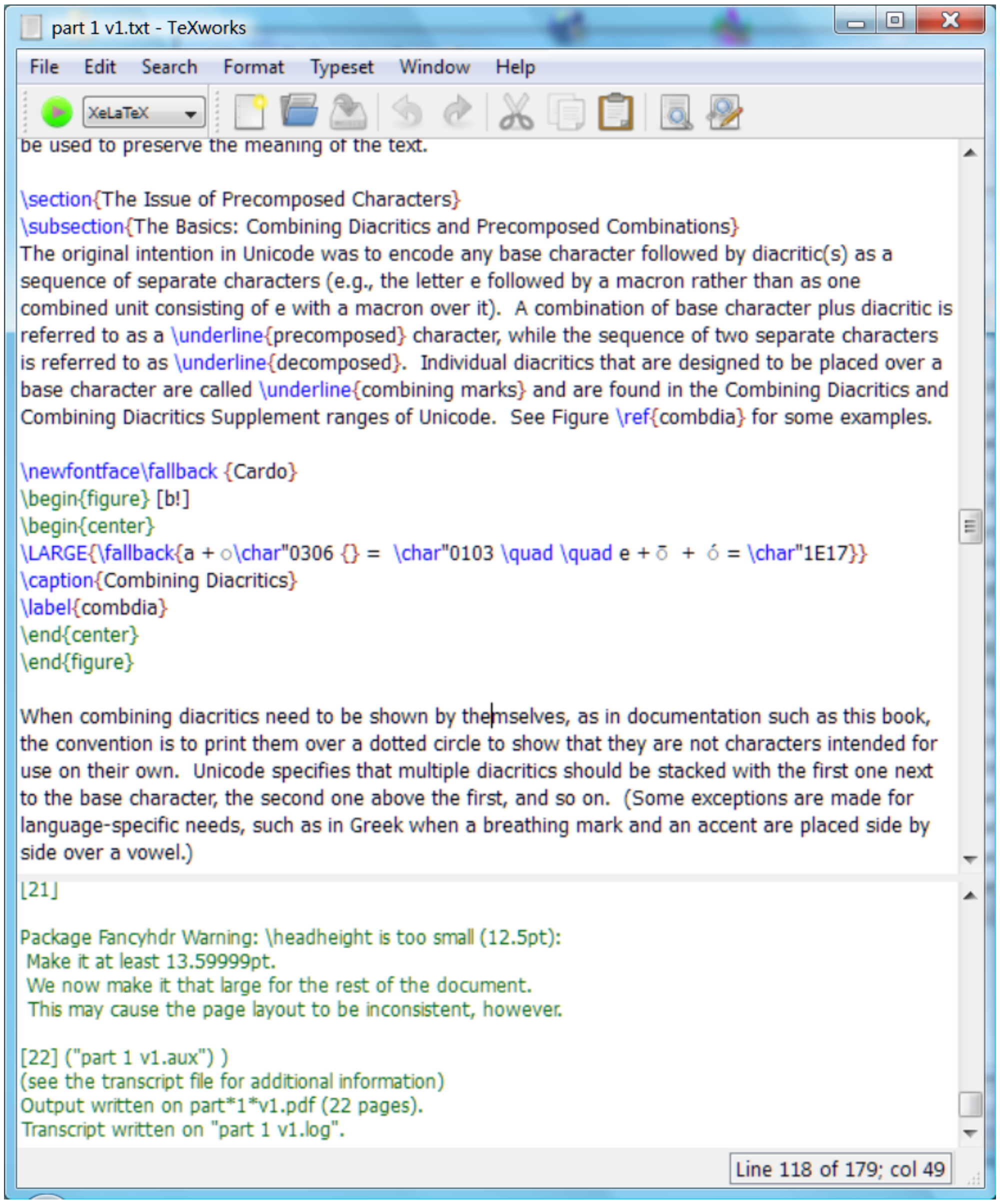
Figure 1: The TEXworks environment.

XƎTEX, given the cost of commercial products and lack of alternatives.

There are also converters available that can change existing documents into TEX and vice versa; one good one for Microsoft Word can be found at http://kebrt.webz.cz/programs/word-to-latex/. OpenOffice Writer (ver. 3.0 or higher) has a built-in converter (File > Export . . . > LaTeX2e). Standalone converters for OpenOffice are also available; see http://writer2latex.sourceforge.net/. Such converters are certainly not perfect, but they can be helpful for beginners or for those who have already invested much time in producing a document in Word or OpenOffice format.

TEX is very powerful. Once you learn how, you can simplify your work in various ways and produce some effects that are difficult or impossible even in sophisticated word processors. TEX and XƎTEX most definitely have a learning curve. The good news is that it is not difficult to get simple text (i.e., text that contains mostly plain paragraphs) working right. But once you start using additional features such as tables, things get a lot trickier. You should approach TEX as you would any other piece of software that requires significant effort to learn. But you will be rewarded with superb-looking documents that take advantage of the most up to date "smart font" technology.

Furthermore, TEX has been around a long time and is widely used by academics and by specialized publishing houses. This, combined with the fact that users can customize TEX by creating packages (extensions to TEX for spe-



```
be used to preserve the meaning of the text.

\section{The Issue of Precomposed Characters}
\subsection{The Basics: Combining Diacritics and Precomposed Combinations}
The original intention in Unicode was to encode any base character followed by diacritic(s) as a
sequence of separate characters (e.g., the letter e followed by a macron rather than as one
combined unit consisting of e with a macron over it).  A combination of base character plus diacritic is
referred to as a \underline{precomposed} character, while the sequence of two separate characters
is referred to as \underline{decomposed}.  Individual diacritics that are designed to be placed over a
base character are called \underline{combining marks} and are found in the Combining Diacritics and
Combining Diacritics Supplement ranges of Unicode.  See Figure \ref{combdia} for some examples.

\newfontface\fallback {Cardo}
\begin{figure} [b!]
\begin{center}
\LARGE{\fallback{a + ○\char"0306 {} =  \char"0103 \quad \quad e + ◌̄  +  ◌́ = \char"1E17}}
\caption{Combining Diacritics}
\label{combdia}
\end{center}
\end{figure}

When combining diacritics need to be shown by themselves, as in documentation such as this book,
the convention is to print them over a dotted circle to show that they are not characters intended for
use on their own.  Unicode specifies that multiple diacritics should be stacked with the first one next
to the base character, the second one above the first, and so on.  (Some exceptions are made for
language-specific needs, such as in Greek when a breathing mark and an accent are placed side by
side over a vowel.)
```

```
[21]

Package Fancyhdr Warning: \headheight is too small (12.5pt):
 Make it at least 13.59999pt.
 We now make it that large for the rest of the document.
 This may cause the page layout to be inconsistent, however.

[22] ("part 1 v1.aux") )
(see the transcript file for additional information)
Output written on part*1*v1.pdf (22 pages).
Transcript written on "part 1 v1.log".
```

Figure 2: A larger view of the editor window.

cific purposes), means that there is some support for scholarly writing with TEX that you will not find with a word processor. For example, there are two packages available that aid in the creation of critical editions of text; complicated numbering schemes and apparatus criticus are much easier to handle with these packages.[3]

3 More useful things to know

- TEX was developed by Donald Knuth in the late 1970s specifically for typesetting books, particularly mathematical ones. It therefore long predates the development of Windows and Mac OS and also Unicode and TrueType/OpenType fonts. The "X" in TEX is pronounced as in Scotish *loch* or German *Buch*. For more history, see the Wikipedia entry at http://en.wikipedia.org/wiki/TeX.
- Many mathematicians use TEX as a standard tool, but it is less widely used in other areas. If you need help with TEX, a math colleague may be a good resource—although he or she may not know anything about the multilingual aspects of TEX.
- TEX is available on Windows, Unix, and Mac OS, which may be important for those who need cross-platform compatibility.
- Over the years three new versions of TEX have been created, building on the original work of Donald Knuth. The various "dialects" of TEX may be summarized as follows:
  - TEX: the original, often referred to as "plain" TEX; very powerful typesetting but not terribly user-friendly and restricted to the old code pages of 256 characters (i.e., no Unicode)
  - LATEX: a more user-friendly version of TEX, containing many commands (macros) that allow users to do things easily that would require complicated programming in plain TEX. LATEX itself has been supplemented with many packages that offer additional functionality in relatively easy to use form. LATEX is currently in its second version, referred to as LATEX2$_\epsilon$. Most LATEX commands and packages are compatible with XƎTEX.
  - XƎTEX: Unicode-based TEX plus the ability to use fonts already installed in the user's system; not particularly user-friendly in its native form and so supplemented with packages such as fontspec. The combination of XƎTEX plus commands and packages originally written for LATEX is referred to as XƎLATEX, which is what most readers of this document will end up using.

[3]The packages are ednotes and ledmac; the former is perhaps easier to use. For more information, see http://www.webdesign-bu.de/uwe_lueck/critedltx.html .

  - ConTEXt is another version, somewhat similar to LATEX in that it is intended to be more user-friendly. Some claim it is better at typesetting, but the choice is really a matter of preference and individual needs. ConTEXt is less widely used than LATEX, so I suggest that beginners focus on learning the latter since there is more help available.

- You may have trouble doing certain things or think they can't be done in TEX. Before giving up, see if someone has written a package that will do what you need. For example, TEX provides a tabular environment that allows you to create simple tables. But this environment can't break tables across pages, print a header row at the top of each page of a multipage table, use color in tables, and do many other things you might want. The packages longtable and colortbl enable you to do them. The need to look for additional packages makes using TEX a bit messier than using a standalone application; it's just part of the learning curve. The best place to look for packages not included with your distribution of TEX is in the CTAN repository (see page 17).

- Unicode support in TEX is relatively new; it began in 2004 when Jonathan Kew released the XƎTEX package, about which more will be said below. From the beginning, TEX users could enter the characters needed for European languages, and later on packages were created to handle other scripts such as Arabic. But these older systems are not Unicode-based and should not be used now for serious multilingual work. Much of the information that you can find online about TEX may be outdated, particularly in regard to support for languages other than English.

- XƎTEX includes support for OpenType features and AAT features.

- Prior to XƎTEX, users could not directly use fonts installed in their operating system. TEX comes with a few fonts, the Computer Modern family designed by Donald Knuth as part of the original TEX system. These fonts, used by default in TEX, have a rather light look to them and it is often possible to recognize a document produced with TEX just by the look of these fonts. It is possible to add other fonts to a TEX installation, but this requires much additional work. Until XƎTEX, lack of support for Unicode and system fonts made TEX an inappropriate choice for the growing number of people who require Unicode-based software.

- At a minimum, you will need three things:
  - A version of TEX, referred to as a *distribution*; in particular, you need XƎTEX if you want to use Unicode and your installed system fonts. A distribution includes many popular packages in addition to the basic TEX typesetting engine.
  - An editor to create the source code; there are editors specifically designed for this task, although any plain text editor can be used as long as it is Unicode-based.

  - A way to view and print the output files; there are various ways to do this, some of which require Ghostscript and GSview (open-source equivalents to Adobe's PostScript language and Reader software).

- There are programs designed to help the beginner get working with TEX as easily as possible. In particular, beginners should take advantage of one of the integrated development environments that aid users in installing TEX, entering and correcting the source code, and producing the final output within one application. At the time this was written, the only such environment on Windows that can handle Unicode and OpenType on Windows is TEXworks, so I will assume that is the environment that you are using. TEXworks is also available for Linux and OS X, and I recommend it on those platforms also, but most Mac and Linux editors are Unicode-capable so you have more choices than on Windows.[4]

In short, if you are ready to try TEX, you need to do four things:

- Download and install the software.
- Learn your way around TEXworks (or whatever editor you are using).
- Start learning LATEX, which will enable you to create basic documents.
- Use XETEX-specific features to control font selection and apply OpenType features.

Note to Mac and Linux Users

Up to this point the discussion about TEX has been quite general and applies to everybody. From this point forward most of the examples will come from Windows. Everything should (theoretically) apply to XETEX when running under Linux. But I do not have a Linux machine or the expertise to test; comments from Linux folks are welcome. Linux users who want to get TEX should obtain the TEXLive distribution from http://tug.org/texlive/.

I will give as much information as I can to help Mac users, although I have not had time to try everything on both platforms. I suggest MacTEX as the best distribution for Mac users; see http://www.tug.org/mactex/2009/, which was current at the time this was written, although a 2010 version will be posted at some point.

There is one caution that must be observed by Mac users who want to create cross-platform documents. Avoid doing things with AAT fonts that will

[4]For the sake of completeness, I will mention some other good Windows software for beginners that I have found, but which does not yet support Unicode and OpenType; such packages may be good choices in the future if they are upgraded. The ProTeX package at http://tug.org/protext/ installs MiKTEX along with TEXnicCenter, a good integrated environment. However, version 1 of TEXnicCenter does not support Unicode; version 2, now under development, will do so. TEXmaker, http://www.xm1math.net/texmaker/index.html, is a good editor that works with Unicode, but it does not yet know anything about XETEX. WinEdt, http://www.winedt.com/, is designed to work with MiKTEX, but has extremely limited Unicode support.

not translate well when implemented with OpenType. The way to do this, if you are not an expert on AAT and OT, is to stick to fontspec and polyglossia for selecting fonts and features and for multilingual support, respectively; these packages are discussed in detail below. The fontspec manual clearly identifies those items (a small number) that are applicable only to Mac OS X. Also, use fonts that are available for all platforms (see Appendix D for font information.) If you do this, your Mac-generated XƎTEX documents should work well on Linux or Windows.

4 Setting Up the Software

The following assumes that your computer system is Unicode-based and that you already have things set up to enter text in whatever languages you require.

4.1 For Windows

- Obtain a TEX distribution. I suggest MiKTEX, but see the third bullet below for an alternative. Download and install MiKTEX from http://miktex.org/. MiKTEX is a widely used distribution of TEX on Windows; it includes XƎTEX and many other packages you need. MiKTEX includes a convenient installer. It also has an excellent feature that tells you if a document requires a missing package and downloads and installs it for you (with your permission, of course).
- MiKTEX puts a program group in Windows's Start menu. You won't have to do much with MiKTEX once it's installed since most of your work will be done in TEXworks (or whatever editor you use). If you wish, you can use the icons in this program group to check for updated packages, change some MiKTEX options, and consult the MiKTEX manual and FAQ.
- The TEXLive distribution, available from http://tug.org/texlive/, is the other option for Windows users. My personal experience is that MiKTEX is easier for beginners, but certainly many people do use TEXLive.
- Both MiKTEX and TEXLive now include the TEXworks editor, which I use and will demonstrate in this article. You can also visit http://www.tug.org/texworks/, which provides a link to the downloads at Google Code (for the technically inclined). Just unzip all the files in the same directory and doubleclick on the TEXworks icon to start the program (there is no installer program).

4.2 For Linux

- You want to use the TEXLive distribution: http://tug.org/texlive/.

4.3 For Mac OS X

- The best distribution is MacTEX, which is TEXLive with some Mac-specific additions. See http://tug.org/mactex/. Note that this is a *very* large download, and there are some options for smaller downloads available; read the notes on the web page carefully. If you get the BasicTEX version, be sure to read *http://www.uoregon.edu/ koch/BasicTeX.pdf*, which gives necessary installation instructions. In addition to the distribution, the MacTEX page also has some good information for beginners.
- MacTEX includes the both the TEXShop editor and TEXworks. The latter is similar to TEXShop but is cross-platform and includes some updated features.

5 Learning TEX works and LATEX

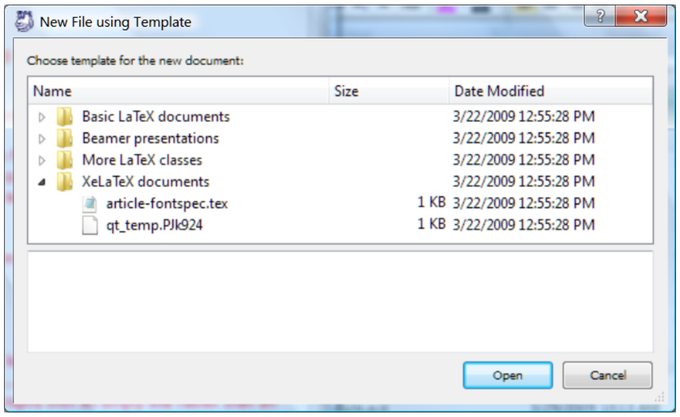
5.1 Document Basics

The "Short manual for TeXworks" by Alain Delmotte, available from http://www.leliseron.org/texworks/ is the place to start. I will mention only a few things to supplement this manual.

Every TEX document has two essential parts, the *preamble* and the *body*. The preamble contains settings that affect the document as a whole (these are not printed in the final output). One can use TEXworks to process source code that is written using any of the variants of TEX (plain TEX, XETEX, LATEX, ConTEXt, etc., as described in section 3). The short sample document shown in section 3.2 of the "Short Manual" is designed to work with LATEX, and its preamble contains a couple of lines that should be revised if one wants to use the advanced font selection features of XELATEX. To create a basic document for XELATEX, choose File/New from Template... and you will see the window shown in Figure 3.

Choose the article-fontspec.tex template and a new document will be created that looks like Figure 4. Note the use of the % character to separate the actual commands from comments. Note also the curly brackets { } that enclose options for the various commands. These are very important; to avoid error messages, don't accidentally delete one or add one too many. The backslash character \ before each command is also essential. If the text in the editor is too small, enlarge it or change to a more legible font using the Format>Font command (this affects what you see in the editor, not what appears in the final output). To make your choice the default whenever you start TEXworks, set it using the Edit/Preferences dialog.

The preamble begins with the \documentclass command. Every TEX document must include this. For this example we will use the article class, which is designed (as its name implies) for writing articles of the sort that appear in academic journals. Following the \documentclass command you find several examples of the \usepackage command. This command tells TEX to include



Name	Size	Date Modified
Basic LaTeX documents		3/22/2009 12:55:28 PM
Beamer presentations		3/22/2009 12:55:28 PM
More LaTeX classes		3/22/2009 12:55:28 PM
XeLaTeX documents		3/22/2009 12:55:28 PM
article-fontspec.tex	1 KB	3/22/2009 12:55:28 PM
qt_temp.PJk924	1 KB	3/22/2009 12:55:28 PM

Figure 3: TEXworks's File / New from Template . . . dialog.

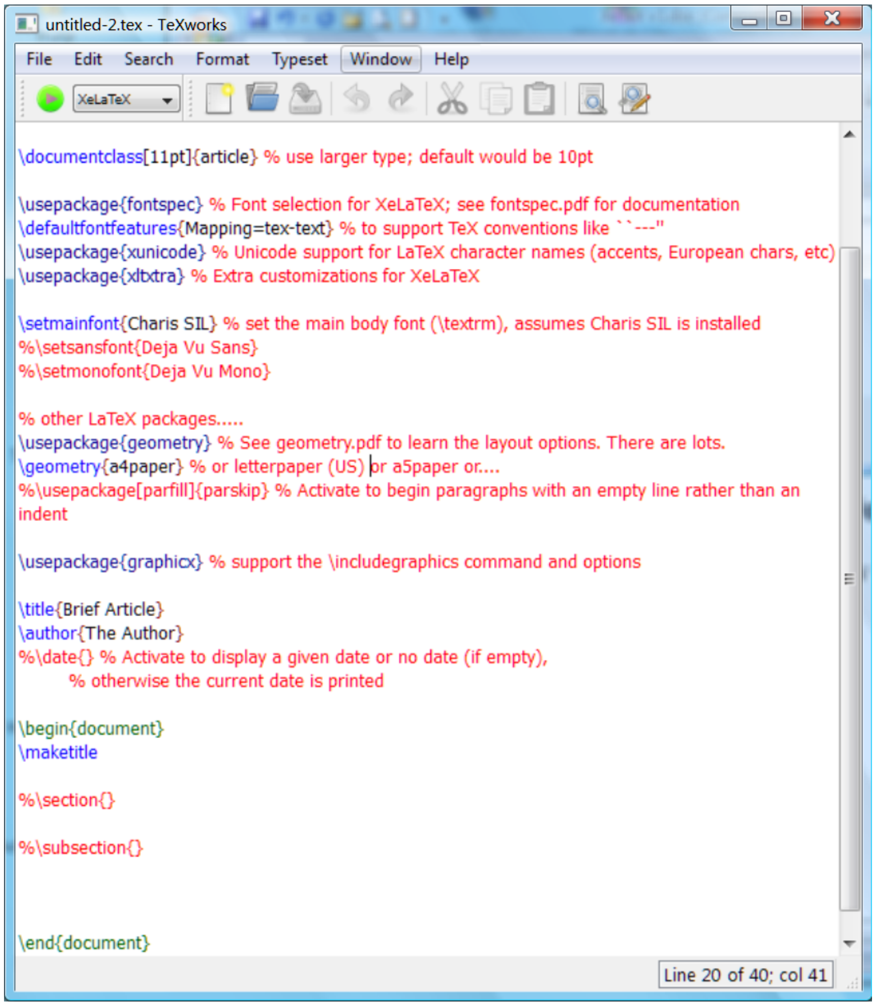
the packages that you need to use when it compiles the document. If you use a command that depends on having a particular package loaded and the package is not given in the preamble, you will get error messages. For XƎTEX documents, always use the fontspec, xunicode, and xlxtra packages. The graphicx package is needed if you want to include any graphics; you can remove it otherwise.

Unless you have the font Charis SIL on your system, change the \setmainfont to something more appropriate. If you are in the US, just delete (or comment out using %) the \geometry package, since TEX defaults to US letter size paper.

The body of the document is located between the \begin{document} and \end{document} commands. (These two commands, along with the initial \documentclass, are the absolute minimum required to set up a TEX document.) Type whatever you want here to create your first document.

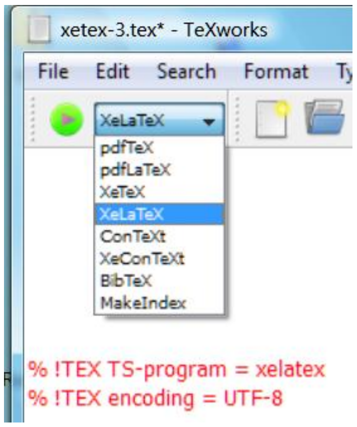
5.2 Typesetting and Error Correction

After you enter some text, you will want to see how the results will look. It is extremely important to choose the right processor before you typeset (TEXworks's term for compiling) the source code. If you include a XƎTEX-specific command but typeset with LATEX, you will get error messages. Open the pull-down menu shown in Figure 5 and select XƎLATEX. This allows you to use both common LATEX packages and commands, along with XƎTEX-specific things like fontspec. (If your document contains only LATEX, it is still OK to typeset it with XƎLATEX.) Then click on the button to typeset your document. You can set the default typesetting option in Edit/Preferences . . . via the Typesetting tab, and I recommend that you do so; that way you can just type CTRL-T when you are ready to view the results. You can also add a line at the very beginning of the


untitled-2.tex - TeXworks
File Edit Search Format Typeset Window Help
XeLaTeX

```latex
\documentclass[11pt]{article} % use larger type; default would be 10pt

\usepackage{fontspec} % Font selection for XeLaTeX; see fontspec.pdf for documentation
\defaultfontfeatures{Mapping=tex-text} % to support TeX conventions like ``---''
\usepackage{xunicode} % Unicode support for LaTeX character names (accents, European chars, etc)
\usepackage{xltxtra} % Extra customizations for XeLaTeX

\setmainfont{Charis SIL} % set the main body font (\textrm), assumes Charis SIL is installed
%\setsansfont{Deja Vu Sans}
%\setmonofont{Deja Vu Mono}

% other LaTeX packages.....
\usepackage{geometry} % See geometry.pdf to learn the layout options. There are lots.
\geometry{a4paper} % or letterpaper (US) or a5paper or....
%\usepackage[parfill]{parskip} % Activate to begin paragraphs with an empty line rather than an
indent

\usepackage{graphicx} % support the \includegraphics command and options

\title{Brief Article}
\author{The Author}
%\date{} % Activate to display a given date or no date (if empty),
        % otherwise the current date is printed

\begin{document}
\maketitle

%\section{}

%\subsection{}



\end{document}
```

Figure 4: A basic TEXworks document.



```
% !TEX TS-program = xelatex
% !TEX encoding = UTF-8
```

Figure 5: Choosing the appropriate processor before typesetting.

file to identify the typesetting program to be used for a particular job; this line is shown in Figure 5 (commented out but still processed by TEXworks).

If typesetting mathematics (one of TEX's traditional strengths) is important to you, note the following. As of July 2010, Unicode math fonts are available, but they are relatively new and not as stable as the traditional TEX math fonts. Some people prefer to stick to the traditional math fonts and typeset using LATEX. But if you need the Unicode support (aside from math) and other things that XeTEX provides, you can typeset with XeLATEX but instruct the program to use the older TEX math fonts. See Chapter 7 for informatiion about using fontspec to do this.

You will probably see some errors displayed in the lower pane of the editor. This can be one of the frustrating things for beginners with TEX; sometimes it's not clear what's wrong or how to fix it. Don't forget that every group introduced by a curly brace must be closed by one. Also, some of the errors that are generated don't really affect the final output. It does no harm to hit RETURN when you are presented with a question mark during compilation. If possible, the process will continue, and you will either see the result of the error or find out that it's an insignificant error. The TEXworks editor displays line numbers at the lower right, and you can use the command CTRL-L to jump to a given line in order to fix an error when one is identified during compilation. One useful thing to keep in mind about errors: start by fixing the first one, since an error near the beginning of a document may generate others that will vanish after the offending first one is fixed.

You cannot yet print the PDF file from TEXworks's output panel; instead, you must open the PDF in Adobe Reader or another PDF viewer, then print. This will be fixed in a future version.

You can now use the "Short Manual" to learn more about using the TEXworks's editor, such as how to use the auto-completion feature to save time when typing TEX commands, and much more.

5.3 Learning LATEX

The remainder of this document will deal specifically with how to use fonts and OpenType features. You will need to learn more about LATEX in order to use TEX effectively. Just keep in mind that much of the information you find about using languages other than English is pre-Unicode. Here are some resources for further study:

- *The Not So Short Introduction to LATEX2e*, by Tobias Oetiker and others. This widely used introduction is included with many distributions of TEX, including MiKTEX, so you probably have it. (One annoyance with MiKTEX is that it creates a folder for itself in C:\Program Files and then adds a great many subfolders and sub-subfolders, making it hard to find the documentation that is in fact provided. Do a search for "lshort" to locate this document.)
- *LATEX*, by Wikibooks contributors; available in PDF or HTML from http://en.wikibooks.org/wiki/LaTeX. This is relatively new and quite useful, I find.
- *Formatting Information: A beginner's introduction to typesetting with LATEX* by Peter Flynn, available at http://mirror.cps.cmich.edu/ctan/info/beginlatex/html. Another good online tutorial.
- "Getting Started with LATEX" by David Wilkins, http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/GSWLaTeX.pdf; a bit old, but still useful).
- *LATEX for Word Processor Users* by Guido Gonzato is helpful for newcomers because it shows the LATEX equivalents for common word processor commands. It is distributed with MiKTEX or can be downloaded separately from here: ftp://tug.ctan.org/pub/tex-archive/info/latex4wp/latex4wp.pdf.
- There are several printed books that teach LATEX; a good one is *A Guide to LaTeX* by Helmut Kopka and Patrick W. Daly (Addison-Wesley; fourth edition 2003). There is also *The LATEX Companion* by Frank Mittelbach, Michel Goosens, and other contributors (Addison-Wesley; second edition 2004). It is somewhat more detailed than the book by Kopka and Daly, particularly in its discussions of various LATEX packages.
- A useful two-page summary of LATEX commands is available at http://www.stdout.org/~winston/latex/latexsheet.pdf

The best place to go for further help is CTAN, the Comprehensive TEX Archive Network, at http://tug.ctan.org/. There is lots of helpful stuff here, including the essential TEX catalog online, http://www.ctan.org/tex-archive/help/Catalogue/bytopic.html, where you can find many additional packages and their documentation along with additional tutorials. Another good source of assistance is the LaTeX Community forum at http://www.latex-community.org/.

6 Creating Multilingual Text

6.1 Entering Unicode Text

Since XeTeX is Unicode-based, you can enter standard Unicode text in the way to which you are accustomed. E.g., if you have Windows set up to handle Greek (among other languages) with a polytonic keyboard, you can switch from Latin script to Greek by using the icon in the system tray (or ALT-LEFT SHIFT) as you usually do. For characters not accessible via any keyboard, you can use the XeTeX command `\char"XXXX`; that is, `\char` followed by a double quote mark and a case sensitive hexadecimal number (four digits, or five for characters beyond the BMP; `02DF` works but `02df` does not). Or copy the character using a utility such as BabelMap or Windows's Character Map (BMP only) and paste it into your source code. If you regularly use Unicode characters not on any keyboard, it is easy to create customized keyboards using Microsoft's Keyboard Layout Creator (Windows) or Ukelele (OS X). If you need more information about how to enter characters or about combining marks (discussed in the next paragraph), see my book *Document Preparation for Classical Languages* at http://scholarsfonts.net.

A word should be said about the package `xunicode`, which you probably want to load when working with multilingual text.[5] This package handles the use of combining accent marks. If you enter a combining accent, `xunicode` will substitute the equivalent precomposed character if one exists in Unicode and is included in the font being used. With most software today, the precomposed forms give better-looking results since many applications do not position combining marks well, at least not in all cases. It also allows one to enter characters from the International Phonetic Alphabet (IPA) using the TIPA keystrokes. Some examples will clarify the various ways to get accented characters.

Precomposed example: e-acute (é)

- type it directly using a keyboard that provides e-acute, such as the US-International keyboard in Windows
- type `\'e`. This is a traditional TeX command for entering an acute accent over a vowel.
- enter 'e' followed by a combining acute. To get this character:
  - use a special keyboard that supports combining marks

[5]The `xltxtra` package automatically loads both `xunicode` and `fontspec`, so there is no need to load them separately. If you need to specify an option when loading fontspec, don't worry; the package will only get loaded once even if both it and xltxtra are loaded in the preamble.

  - copy and paste it from a utility such as Character Map or BabelMap
  - use the XeTeX command `\char"0301`

In all cases except the last, your output file will contain the character U+00E9, LATIN SMALL LETTER E WITH ACUTE; `xunicode` has taken care of changing the given text into the appropriate precomposed Unicode character. Combining marks entered using `\char"` plus hexadecimal digits, however, are not changed into precomposed forms.[6]

Most people will prefer the first option (typing the character directly) if it is available, since it lets you see the actual character in your source file. However, all three work equally well. The traditional TeX commands for accent marks are particularly convenient for those who have been working with them for a long time. In addition, they are useful for marks that are usually not supported by keyboards, such as the macron (\=) and the breve (\u), or for characters that you enter only infrequently. Anyone who regularly types in Spanish, for instance, would certainly want to install a keyboard that provides easy access to the necessary characters. Someone who on rare occasions types a Spanish word with accents or inverted question marks might be satisfied with the older TeX conventions. A list of the traditional TeX keystrokes for accents, characters, and common symbols is given in Appendix E.

Example requiring combining marks: y̆

- The combination y-breve does not exist in Unicode in precomposed form, so the only way to enter it is with U+0306, COMBINING BREVE.
- Enter a ‘y’ followed by a combining breve.
  - Enter this character directly if you have a special keyboard that supports combining marks.
  - Copy and paste it from a utility such as BabelMap, Character Map (Windows) or Character Palette (OS X) .
  - Enter it with the command `\char"0306`.
- How well the combination looks on the page will depend on the font in use. Some fonts are designed to position combining marks well over most base characters, while with other fonts the accent may be off-center or, in the case of capital letters, not raised high enough. For example, on my system the Georgia font does not handle the y-breve combination well: Georgia y ̆.
- If you must use combining marks in a font where they don’t all work well, you can manually kern them. When I give the command `y\kern-4pt\char"0306` while working with Georgia, I get a much better result: y̆. The `\kern` command, when followed by a negative value, moves the

[6]If you are creating a PDF to be printed and it looks good, you don’t need to worry about exactly what characters are in the output file. However, if the text may be reused for other purposes later, this information may be important.

following character closer to the previous one.[7] If you need to use such a combination many times in your document, you can define a custom command for it, such as the following:

```
\newcommand{\ybr}{y\kern-4pt\char"0306}
```

which will produce a y-breve every time you enter `\ybr`.

6.2 Old Habits Die Hard

If you look at pre-XƎTEX tutorials or templates, you may be told to add the following to the preamble:

```
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

The first of these is not needed when compiling documents with XƎLATEX, since XƎTEXdefaults to UTF-8.[8] The fontenc package, which specifies whether the fonts in a document use the original TEX character set or an updated one established in 1990, is also unnecessary since XƎTEX automatically uses Unicode fonts.

6.3 Using polyglossia for Additional Language Support

You can enter Unicode text as explained above. However, complete support for a given language may require other things, such as rules about hyphenation, punctuation, date format, and numeral format. Older versions of TEX used a package called babel to support various languages. Ignore any info about babel you find and instead add the newer polyglossia package (provided as part of MiKTEX and other distributions) to your preamble.[9] Using polyglossia you can do a number of things, such as set a default language, enable various additional languages, associate particular fonts with scripts, and more.

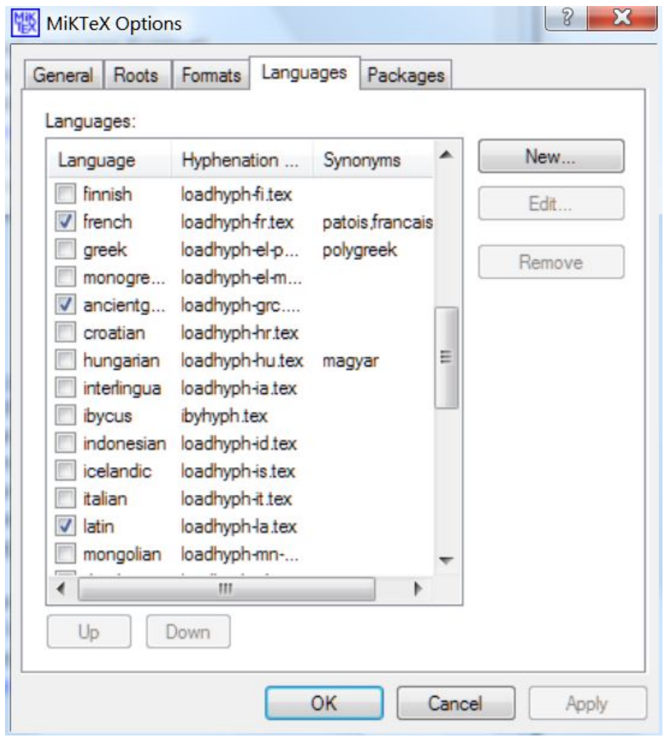
The essential steps are:

- Use the MiKTEX Options control to make sure that the hyphenation files for the language(s) you need will be loaded. From the Start menu, go to All Programs > MiKTEX 2.7 > Settings, then choose the Languages tab and click in the appropriate checkboxes. See the screenshot in Figure 6.
- Load the package polyglossia in the preamble.
- Define the default language and any others that you need.
- Associate fonts with particular scripts and languages, as needed.

[7] Don't put a space between the base letter and the following combining mark, or you will have to kern by a very large amount to back up over the blank space.

[8] UTF-8, short for Unicode Transformation Format-eight bit, is one of several formats that enable Unicode characters to be stored efficiently in data files. For details about the various UTFs, look on the Unicode website, www.unicode.org.

[9] If you ever read the log file that is created when you typeset a document, you may notice a reference to hyphenation patterns with babel. Don't worry about that.



Languages:

	Language	Hyphenation ...	Synonyms
☐	finnish	loadhyph-fi.tex	
✓	french	loadhyph-fr.tex	patois,francais
☐	greek	loadhyph-el-p...	polygreek
☐	monogre...	loadhyph-el-m...	
✓	ancientg...	loadhyph-grc....	
☐	croatian	loadhyph-hr.tex	
☐	hungarian	loadhyph-hu.tex	magyar
☐	interlingua	loadhyph-ia.tex	
☐	ibycus	ibyhyph.tex	
☐	indonesian	loadhyph-id.tex	
☐	icelandic	loadhyph-is.tex	
☐	italian	loadhyph-it.tex	
✓	latin	loadhyph-la.tex	
☐	mongolian	loadhyph-mn-...	

Figure 6: Loading hyphenation files.

- Put large amounts of text inside an environment defined by `\begin{`*`language`*`}` and `\end{`*`language`*`}`, where *`language`* is one of the languages you have defined.
- For small bits of text, you can use the command `\text`*`language`*`{ … }`; put the text you want in the specified language inside the curly brackets.

See the sample code in the second half of Appendix C (Section 10.2). Note that hyphenation files for Latin and ancient Greek are not loaded by default, so add them if you wish to typeset this sample. For more details, including the options available for various languages, see the polyglossia documentation; it is not difficult. Read polyglossia.pdf or compile polyglossia.tex; also open the file examples.tex found in the polyglossia folder to study the header. One thing to note: §2.1 of polyglossia.pdf says "You can determine the default language… " which almost sounds like setting a default language is optional. In fact, you need to set the default language in order for things to work properly.

Note that you do not have to use polyglossia if you don't need the special features it provides. The opening of the *Odyssey*, printed below, appears just fine, since the lines of poetry avoid any need for hyphenation and the default font used for this document contains Unicode Greek characters as well as Latin:

> Ἄνδρα μοι ἔννεπε, μοῦσα, πολύτροπον, ὃς μάλα πολλὰ
> πλάγχθη, ἐπεὶ Τροίης ἱερὸν πτολίεθρον ἔπερσεν·
> πολλῶν δ’ ἀνθρώπων ἴδεν ἄστεα καὶ νόον ἔγνω,
> πολλὰ δ’ ὅ γ’ ἐν πόντῳ πάθεν ἄλγεα ὃν κατὰ θυμόν,
> ἀρνύμενος ἥν τε ψυχὴν καὶ νόστον ἑταίρων.

(The source code for this document does load polyglossia, but I did not put `\begin{greek}` and `\end{greek}` tags around these lines, so they get no special treatment.)

I think that polyglossia provides the easiest and most consistent way to use different languages in XƎTEX, and I definitely recommend it for beginners. If you choose not to use polyglossia, you may need to take one additional step when dealing with complex scripts. If you find that the correct characters are present in your text but are not shaped properly, make sure that your specify the script when you give a fontspec command, like this:

```
\fontspec[script=devanagari]{Code2000}
```

where the name of the script you are using is given as an option between square brackets. This will enable the proper shaping. (Don’t do this if you are using a native AAT font on Mac OS X.)

6.4 Creating Right to Left Text

You should not have trouble entering right-to-left text if you already have your computer set up to use Hebrew, Arabic, or another of the right to left scripts that are supported in Unicode. (If you have not yet set up that capability, you need to do so before using RTL text in XƎTEX.) The following examples will use Hebrew, but the same principles will apply to Arabic, Syriac, etc.

6.4.1 Using polyglossia for RTL

Using polyglossia lets you do three things easily:

- switch language and script with one command and automatically activate the OpenType shaping features needed for the language
- associate fonts with particular scripts or languages
- access other options (calendar style, numeral style, locales for Arabic)

General information about polyglossia is found beginning on page 20 above. See that section for directions on loading polyglossia and associating scripts or languages with particular fonts.

For small amounts of right-to-left text within a left-to-right environment (or vice versa), use the command `\text`*`language`*`{ … }` , replacing *language* with the language you want and putting the desired text between the curly brackets. For full paragraphs, put your right-to-left text inside an environment

(\begin{Hebrew} . . . \end{Hebrew}). Either of these methods gets words in RTL text to appear in the correct order, and using the environment will get your paragraphs aligning properly at the right (not left) margin.

polyglossia calls an associated package bidi. Make sure that you have the most recent version of bidi, at the time this was written, the current version was 1.1.4c, which resolves a number of issues from earlier versions. If you work extensively with any language that uses RTL text, you should download the bidi manual from CTAN; it is easy to read and provides much more information that I can include in this short tutorial. Originally written by François Charette, bidi is now maintained by Vafa Khalighi who has done much to improve and extend this important package.

6.4.2 Using bidi by itself

If you do not need any of the special features that polyglossia provides elsewhere in your document, you can load the bidi package directly in your preamble (and omit polyglossia). If using bidi by itself, note the following:

- Be sure to load bidi after all other packages, except xunicode which should come after bidi. (bidi will generate an error message if you load packages in the wrong order.) If your document is mostly or entirely RTL, use the [RTLdocument] option when loading the package — this automatically sets you up for RLT typesetting.
- You must declare the script when using a fontspec command. For instance, to switch from your Latin-script font to the Scheherazade font for Arabic, enter this:

  ```
  \fontspec[script=arabic]{Scheherazade}
  ```

  If you don't do so, the Arabic letters will appear but in the wrong forms (word-initial forms, word-final forms, etc. will not be implemented). On Mac OS X, declare the script if using a Windows-style TrueType/OpenType font; this activates the ICU renderer automatically. If using a native AAT font, do not specify the script.
- bidi provides both commands (\setRTL and \setLTR) and an environment \begin{RTL} . . . \end{RTL} for direction switching. Of course, you should reverse the order of these commands if appropriate.
- For small amounts of text (a few words), use the commands \RLE{ } and \LRE{ }[10]
- See the bidi documentation for complete information about which LaTeX packages bidi supports and about additional options not covered here (footnotes, multiple columns, and other things).[11]

[10] Those who are familiar with the Unicode bidirectional algorithm can easily remember these are Right to Left Embedding and vice versa.

[11] Due to limitations of XeTeX, bidi doesn't work with the color or xcolor packages if you need to color text that takes up more than one line (use xecolour for longer segments).

6.4.3 Special Issues with Unicode's Old Italic Characters

The Old Italic characters are used both for LTR and RTL languages. Unicode decided to define them as strongly LTR, which means that you must take some additional steps if you wish to use them to display an inscription in right to left order. You must put a special Unicode character called RIGHT TO LEFT OVERRIDE, U+202E, before each word when using bidi. The RLO forces characters to behave in the directionality opposite to that which is inherent in them; in this case the strongly LTR charcters will appear as RTL. For one word, enter the RLO followed by the Old Italic characters. For more than one word, you must surround the text with `\RLE{ }` or use the `\setRTL` command (to avoid the problem of words appearing in inverse order) AND precede each individual word with a RLO.[12] For much more information about using Old Italic in RTL mode successfully, see my book on document processing for scholars.

You can insert the Unicode formatting characters such as right to left override, pop directional formatting, etc., by right-clicking in TeXworks's editor window. However, these characters are invisible and you may find it easier to edit the text if you can see them in the source code. To do this, use the XeTeX command `\char"202E` for a RLO and `\char"202C` for a PDF.

7 Using Fonts and OpenType Features

7.1 LaTeX Font Basics

There are a few things you should know as you begin working with fonts. fontspec, following in the footsteps of LaTeX, thinks in terms of font families. We saw on page 14 above how to use the \setmainfont command to set the default font for the document. If the font you specify here has bold, italic, and bold italic versions (and the font maker has named them in the customary ways internally) then the right font will automatically be applied when you ask for bold or italic text. The main font is usually a serifed face. TeX traditionally defines two other families, a sans-serif and a monospaced font; use the \setsansfont and \setmonofont commands. You can ignore these if you don't plan to use the other font families in your document. Notice also that the default font size is usually set as part of the \documentclass command in the preamble.

Aside from the default, font sizes are usually set in relative, not absolute, terms with the LaTeX commands (from smallest to largest) `\tiny`, `\script-`

[12] Those with some experience in using Unicode RTL text may realize that this procedure does not follow the Unicode bidirectional algorithm. This algorithm requires only a RLO at the beginning and a pop directional format (PDF — not to be confused with Adobe's Portable Document Format) at the end for short amounts of text (full paragraphs do require an additional, higher-level command in a word processor or editor to get paragraphs right-aligned.) The need for the additional RLO is due to the way XeTeX interacts with the ICU text renderer that it uses.

size, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge and \Huge. The advantage of this procedure is that if you change the default, LaTeX can adjust all the other font size changes relative to the new default; you don't have to go around manually changing each one. (You can, of course, specify exact sizes if you wish.) The same is true for setting the font families, where one setting at the beginning can control the entire document.

In LaTeX, the appearance of text may be changed in two main ways, by using a declaration or a command. A *declaration* is a command that affects the entire document forward from the point where it is given (unless specifically changed by a subsequent command).

Font changing commands have both command and declaration forms. To switch from the default roman font to the sans-serif family, you can issue either the declaration `\sffamily` or the command `\textsf{sometext}`, where *sometext* is the text that you wish to be printed in the sans-serif font. You can see that if the sans-serif text is long, it would be easy to get lost and forget the closing curly bracket. Furthermore, your italic text might have other font changing commands nested inside it, such as commands for boldface or underlining, potentially leading to further confusion. So, in general, use the declaration form for anything longer than a few words or a short sentence.[13] The table below lists the common font commands.

COMMAND	DECLARATION	EFFECT
\textrm{text}	\rmfamily	applies the roman family
\textsf{text}	\sffamily	applies the sans-serif family
\texttt{text}	\ttfamily	applies the monospaced family
\textnormal{text}	\normalfamily	applies the default family
\textbf{text}	\bfseries	produces **boldface text**
\textit{text}	\itshape	produces *italic text*
\textsc{text}	\scshape	produces SMALL CAP TEXT
\emph{text}	\em	produces *emphasized text*
\underline{text}		produces underlined text

Some notes about font commands:

- *tt* stands for teletype or typewriter, going back to the days when typewriters could produce only monospaced text.
- The word "series" in the term `bfseries` takes into account bold italic as well as upright boldface.
- The command for emphasis is distinct from the command for italics. The latter is used for things that should always be italicized, such as book titles. In most writing styles italics are also used for emphasis, but a document might define a different typographic handling for text that

[13]It is possible to put the declaration form inside a grouping, like this: `{\rmfamily sometext}`. In very complicated situations you may need to do so, but it is unusual.

the author wishes to emphasize. In such a situation the custom style would be applied to any text marked for emphasis. If italics are used, LaTeX takes care of adding any needed space after the last italicized word. If the surrounding text is already in italics, the `\emph` command puts the emphasized text in roman—pretty neat!

- The command for underlining does not have a declaration form because one almost never would want to set a whole paragraph, for instance, as underlined; by its nature underlining is limited to short bits of text.

7.2 About fontspec

fontspec is a package written specifically to work with XeTeX (so you can't use it if you compile your document with plain LaTeX). It is designed to let you choose fonts easily, control OpenType features, and do many other things. You definitely want to put it in your preamble. Note to experienced TeX users: fontspec is the best and easiest way to select fonts in XeTeX; the `\font` command does not behave the same way as it does in other TeX variants, and you should avoid it unless you really know what you are doing.

fontspec was originally developed on the Mac and some of the names it uses to call features are typical of AAT rather than OT fonts (e.g., Uppercase rather than Lining numerals). Appendix A contains a table that will help you sort out the names. You may know the OT names from experience with applications such as InDesign or from reading documentation that comes with your OT fonts. In a few cases you can use either name, as the table shows. The integration of AAT and OT features within one framework is an excellent feature of fontspec—particularly now that Mac OS X is processing many OT features—and the need to sort out the names for those accustomed to OT is a small price to pay.

7.3 fontspec commands

fontspec commands take the following form:

```
\fontspec[font features]{font name}
```

So if you included this command in your preamble:

```
\fontspec[Ligatures=Discretionary,Numbers=OldStyle]{Linux Lib-
ertine}
```

it would set the font to Linux Libertine with the OT Discretionary Ligatures and Oldstyle figures turned on. Note the following:

- The OT (or AAT) features go inside square brackets.
- The name of the feature is followed by an equal sign, then the option; multiple features may be selected at once if separated by a comma; no spaces allowed inside the square brackets.

- Names are case sensitive.
- See the chart in Appendix A for a list of OT features and options plus some additional information.
- Names of fonts inside the curly brackets must be the display name, i.e., the name shown when you pull down an application's font menu (it may contain spaces).

You will want to set any OT features you want to apply throughout the document as part of the preamble.

To add a new OT feature to the font and features already in effect, you can use the `\addfontfeatures` command (or `addfontfeature` for one feature). Note that there is a slight difference in syntax between `\addfontfeatures` and other fontspec commands:

```
\fontspec[Numbers=OldStyle]{Cardo}
\addfontfeature{Numbers=OldStyle}
```

The first example applies both a font change and an optional feature, while the second applies only a different feature; note the use of square brackets in the first example. In TEX, some commands consist only of a backslash followed by one word, such as `\smallskip` to insert some vertical space. Other commands may require or accept additional information; such specifications are called arguments. Required arguments go in curly brackets, while optional arguments are put between square brackets and are located between the command and the required arguments. A `\fontspec` command requires the name of a font, for obvious reasons, and so the font name goes inside curly brackets, with options inside square brackets. The `\addfontfeature(s)` command requires the name(s) of the additional feature(s), so curly brackets are used.

Do note that any changes made with \addfontfeatures will apply to the rest of the document or until they are cancelled by a new command. If you want to apply a feature to only a small amount of text, put the entire command and the associated text inside curly braces like this:

```
{\addfontfeatures{Numbers=OldStyle}
In 1894 and again in 1902 we went West.}
```

Here the oldstyle numerals will be applied only to this one short sentence, and the default lining numerals will resume after the end of the group. If the text is long or you get lost in too many curly braclets, you can create an environment using \begin and \end commands; see the standard LATEX tutorials for more about this.

If you wish to have a particular font associated with a specific script or language, see section B.3 on page 34. You may need to do this if, for instance, the main font you are using in a document does not support Greek or Hebrew and you need to use those languages in your document.

Appendix B provides a summary of the most common fontspec commands. For more examples, see the short code sample in Appendix C. This sample is designed to help you learn to control fonts and OT features, not to demonstrate many general features of LaTeX (there are other places to look for that!). Remember that the `%` character sets off comments from the actual commands, and I have included comments to help you undestand what each line does. You can copy this code, paste it into TeXworks, and typeset it on your own system (change the Linux Libertine font if you don't have it, or get it from http://linuxlibertine.sourceforge.net/. It's freely available and despite its name works on Windows or Mac as well as Linux. Appendix D lists some other freely available fonts that contain OT features for you to experiment with (not all the fonts contain exactly the same features, so check the documentation that comes with each). Also be aware that a font with a TrueType extension (.ttf) can contain OT features; check the information provided by the font maker if you are unsure.[14] You can also run the XeTeX macro *opentype-info.tex* on a font and get a PDF file listing the OT features present in the font.[15]

7.4 Learning More

MiKTeX includes fontspec documentation in PDF form. It is well written but gets rather technical at times (as it must), and it is particularly difficult for those who are new to TeX; those already experienced with LaTeX will have an advantage. If you have understood the material in this article, studied the sample code at the end, and done some experimenting, you will then be ready to read the actual fontspec manual. Let me mention a couple of things you can learn there as examples of the power of XeTeX and fontspec.

fontspec includes the \scale command to match the heights of lowercase (or uppercase, if you wish) characters from different fonts. If you have ever mixed, e.g., Times New Roman and Arial, you've noticed that the Arial characters seem a little too big relative to Times even at the same point size, and you had to manually change each bit of text in Arial to be a couple of points smaller. fontspec can handle that for you. Some typefaces come in a number of different weights in addition to the regular and boldface: light, book, demi, heavy, black and so forth. You can tell fontspec to use the demi version as the "boldface" when the light version is the main font; in other words, you can define your own font families. You may not need to do this often, but when you need it, this ability is very helpful.

Have fun!

[14]Fonts that have the .otf extension may contain outlines in the Compact Font Format (CFF), an update of the older Postscript Type 1 format, or they may contain TrueType outlines. Windows recognizes TrueType fonts that have a digital signature as OpenType, whether or not they contain advanced typographical features. TrueType fonts without the digital signature but with advanced features still get the extension .ttf.

[15]Use *aat-info.tex* for AAT fonts on Mac OS X.

A Appendix: Fontspec and OpenType Features

In the table on the following pages:

- The lefthand column is the fontspec feature name (given in the same order as in the fontspec documentation).
- The second column, Fontspec Feature Options, shows the tags used to turn various options on and off. The names are case-sensitive, i.e., `OldStyle` works but `Oldstyle` does not.
- These tags work with either the `\fontspec` or the `\addfontfeatures` command, e.g.:
  `\fontspec[Letters=SmallCaps]{MinionPro}` or
  `\addfontfeatures{Ligatures=Discretionary,Historical}`
- The OpenType Name and the four-letter OpenType tag are provided for those who have some familiarity with OT features outside of fontspec and want to match the names they know with those used by fontspec. For example, if you have used Adobe InDesign or read the official OT specification, you are familiar with Tabular Figures, not Monospaced. Ignore these if they don't mean anything to you.
- The righthand column indicates features that should be turned on by default with a double asterisk. Not all applications (including Microsoft's own Office suite) follow This part of the OT spec, at least not completely.
- This table omits OT features that are applicable only to Arabic, East Asian languages, etc.
- Note that "Alternate" is an option within the Contextuals, Fractions, and Style features and is also the name of a separate feature. See the notes below for clarification.

Of course, the font in use must support the feature(s) you want to apply.

The table begins on the next page and continues onto page 32.

FONTSPEC FEATURE	FONTSPEC FEATURE OPTIONS	OPENTYPE NAME	OT 4L	ON BY DFLT?
OpticalSize[16]	*= point size*	Optical Size	`size`	
Ligatures	Required	Required Ligatures	`rlig`	**
	Common	Standard Ligatures	`liga`	**
	Discretionary (*or* Rare)	Discretionary Ligatures	`dlig`	
	Contextual	Contextual Ligatures	`clig`	**
	Historical	Historical Ligatures	`hlig`	
	TeX	*(not an OT feature)*	`tlig/trep`	
	Turn Ligatures off by prefixing the option with No, e.g., `NoHistorical`			
Letters	Uppercase	Case-Sensitive Forms	`case`	
	SmallCaps	Small Capitals	`smcp`	
	PetiteCaps	Petite Capitals	`pcap`	
	UppercaseSmallCaps	Small Capitals from Capitals	`c2sc`	
	UppercasePetiteCaps	Petite Capitals from Capitals	`c2pc`	
	Unicase	Unicase	`unic`	
Numbers	Monospaced	Tabular Figures	`tnum`	
	Proportional	Proportional Figures	`pnum`	
	OldStyle (*or* Lowercase)	Oldstyle Figures	`onum`	
	Lining (*or* Uppercase)	Lining Figures	`lnum`	
	SlashedZero *Turned off by* NoSlashedZero	Slashed Zero	`zero`	
	Arabic	*(not an OT feature)*	`anum`	
Contextuals	Swash	Contextual Swash	`cswh`	
	Alternate	Contextual Alternates	`calt`	**
	WordInitial	Initial Forms	`init`	
	WordFinal	Terminal Forms	`fina`	
	LineFinal	Final Glyph on Line Alternates	`falt`	
	Inner	Medial Forms	`medi`	
	Turn Contextuals off by prefixing the option with No, e.g., `NoSwash`			

[16] If you have an OT font that comes with different optical size versions, fontspec will automatically select the appropriate one based on the font size specified in the document, so this command is normally not used. You can adjust optical size if desired with the `OpticalSize` command or turn it off with `OpticalSize=0`.

FONTSPEC FEATURE	FONTSPEC FEATURE OPTIONS	OPENTYPE NAME	OT 4L	ON BY DFLT?
VerticalPosition	Inferior	Subscript	`subs`	
	Superior	Superscript	`sups`	
	Ordinal	Ordinals	`ordn`	
	ScientificInferior	Scientific Inferiors	`sinf`	
	Numerator	Numerators	`numr`	
	Denominator	Denominators	`dnom`	
Fractions	On (turned off by Off)	Fractions	`frac`	
	Alternate	Alternative Fractions	`afrc`	
StylisticSet	1 *or* 2 *or* 3 . . . 20; *see note 17*	Stylistic Set	`ss01–ss20`	
CharacterVariant	1 *or* 2 *or* 3 . . . 20; *see note 17*	Character Variants	`cv01–cv20`	
Alternate	0 *or* 1 *or* 2 *or more numbers; see footnote*[17]	Stylistic Alternates	`salt`	
Style	Alternate[18]	Stylistic Alternates	`salt`	
	Italic	Italics	`ital`	
	Swash	Swash	`swsh`	
	Historic	Historical Forms	`hist`	
	TitlingCaps	Titling	`titl`	
	Ruby	Ruby[19] Notation Forms	`ruby`	
	HorizontalKana	Horizontal Kana Alternates	`hkna`	
	VerticalKana	Vertical Kana Alternates	`vkna`	
Kerning	On (*turned off by* Off)	Kerning	`kern`	**
	Uppercase	Capital Spacing	`cpsp`	
Artificial font transformations are likely to produce poor quality typesetting, so use only as a last resort.				
FakeSlant	0.1, 0.2, 0.3 etc.	*(not an OT feature)*		
FakeStretch	1.1 *or greater*	*(not an OT feature)*		
FakeBold	1.1 *or greater*	*(not an OT feature)*		

[17] Stylistic Alternates are selected numerically; e.g., `Alternate=2`; the default is numbered `0`. Stylistic sets, however, begin with `1`. See your font's documentation for information about the stylistic sets, character variants, and stylistic alternates it supports.

[18] Simply turning on `Style=Alternate` will select the first variant, if more than one is defined in `salt`. To access additional alternates (second, third, etc.), use `Alternate=2, Alternate=3` and so forth, as defined in the Alternate feature.

[19] This and the next two features are used in Chinese, Japanese, and Korean (CJK) typesetting. See the next page for more CJK features.

FONTSPEC FEATURE	FONTSPEC FEATURE OPTIONS	OPENTYPE NAME	OT 4L	ON BY DFLT?
Items on this page are used for CJK typesetting. *See also the Ruby, HorizontalKana and VerticalKana features on page 31.*				
Annotation		Alternate Annotation Forms	`nalt`	
CJK Shape	Traditional	Traditional Forms	`trad`	
	Simplified	Simplified Forms	`smpl`	
	JIS1978	JIS78 Forms	`jp78`	
	JIS1983	JIS83 Forms	`jp83`	
	JIS1990	JIS90 Forms	`jp90`	
	Expert	Expert Forms	`expt`	
	NLC	NLC Kanji Forms	`nlck`	
Character width	Proportional	Proportional Widths	`pwid`	
	Full	Full Widths	`fwid`	
	Half	Half Widths	`hwid`	
	Third	Third Widths	`twid`	
	Quarter	Quarter Widths	`qwid`	
	AlternateProportional	Proportional Alternate Widths	`palt`	
	AlternateHalf	Alternate Half Widths	`halt`	

B Appendix: fontspec Command Summary

This is meant as a "cheat sheet" that you can keep handy while working. You may need to study the examples in fontspec.pdf to see how these options really work. A few of the rarer fontspec commands are not listed here.

Remember that all fontspec *commands are case-sensitive.* So `BoldItalic` works but `Bolditalic` will generate all sorts of error messages.

B.1 Basic Format

`\fontspec[`*`features`*`]{`*`font name`*`}`

This is the general form of fontspec commands. *`features`* is replaced by one or more features, or it may be omitted (in which case don't type the square brackets). Multiple features may be separated by a comma. The *`font name`* must be the display name, i.e., the name shown when you pull down the font menu in an application, which may contain spaces. This first examples changes only the font:

```
\fontspec{Linux Libertine}
```

while this second example selects a new font and applies two features:

```
\fontspec[Numbers=OldStyle,Color=0000FF]{TeX Gyre Pagella}
```

B.2 Additional Commands

`\setmainfont[`*`features`*`]{`*`font name`*`}`
`\setsansfont[`*`features`*`]{`*`font name`*`}`
`\setmonofont[`*`features`*`]{`*`font name`*`}`
`\defaultfontfeatures{`*`features`*`}`

These four commands are usually used in the preamble; the first one sets the roman font family. Features can be set individually for the various families, but if you want to set defaults for them all you can use `\defaultfontfeatures`.

`\addfontfeatures{`*`features`*`}` (this can also be called as `\addfontfeature`)

This command adds additional feature(s) to those already in effect, without changing the font. It will continue in effect through the rest of the document unless the entire command and the text it affects are enclosed in curly brackets, like this:

```
{\addfontfeatures{Numbers=OldStyle} 0123456789}
```

or unless it is explicitly turned off by a command later in the document.

Note that there is a difference between the way most font commands are invoked versus `\defaultfontfeatures` and `\addfontfeatures`. In most cases the features come first inside square brackets, followed by a font name in curly brackets. With these two commands there are never any square brackets, and the features are in curly brackets.

Experienced TeX users understand why this is so. New users should note that a TeX command may take a required argument inside curly brackets, preceded by optional argument(s) in square brackets. It would make no sense to give a `\fontspec` command without specifying the font you want, or a `\addfontfeature` command with no features to be applied; such required arguments must go inside curly brackets.

`\addfontfeatures` overrides features called by `\fontspec`, which itself replaces features called by `\defaultfontfeatures`.

`\newfontfamily\`*`name`*`[`*`features`*`]{`*`font name`*`}`

This command sets a font family that can be reused later in the document.

`\newfontface\`*`name`*`[`*`features`*`]{`*`font name`*`}`

This command is similar to the above, but is used for fonts that do not belong to a family, as with some decorative and symbol fonts; automatic selection of bold and italic is not available and presumably not required.

Don't use `\newfontfamily` and `\newfontface` unless you need to reuse the commands multiple times; for an occasional font change, just use `\fontspec`. Note that you must supply your own name for these two commands, shown above as `\`*`name`* between the command and the square brackets for options. So this command:

```
\newfontface\cal[Contextuals=Swash]{Flourishy Font}
```

would allow you to switch to your favorite calligraphic font (hence the shorthand name `\cal`) with contextual swashes turned on simply by typing `\cal`.

B.3 Associating Fonts with Scripts or Languages

You can also use the `\newfontfamily` command if the default font in a document does not support a particular script or language or if you prefer the appearance of a different font for a certain script. The following example sets the font GFS Porson to be used for Greek:

```
\newfontfamily\greekfont{GFS Porson}
```

The name that you give this type of command must begin with a script or language that has been enabled in the preamble using the polyglossia package, as explained on page 20, and must end with the letters `font`: so `greekfont`, `hebrewfont`, `arabicfont`, etc., are valid.

If you are not using polyglossia and you issue a fontspec command to switch to a font that supports a complex script, use this format:

```
\fontspec[script=bengali]{bangla}
```

If you omit the `[script= ]` option, the font will be used but the shaping needed for (in this case) Bengali will not be applied, even though the font supports it.

(This is actually not a fontspec issue, but rather a requirement of XƎTEX, but it seemed useful to mention it here.) This is not required for simple scripts (Latin, Greek, Cyrillic, etc.).

B.4 Features Applicable to Any Font

Items discussed in this section are features. They are not commands in themselves but will appear between square brackets in a \fontspec command or between curly brackets after \addfontfeatures.

BoldFont
ItalicFont
SmallCapsFont

These three features allow you to create your own font families. This is particularly useful if you are using a family that has more than the standard four weights. For example, if you set text in Art Deco Light, you could automatically use the Demi version as the "boldface" by giving the following command:

```
\fontspec[BoldFont={Art Deco Demi}]{Art Deco Light}
```

If you have an older set of fonts which provides small caps in a separate font file, you can get the small caps to be applied automatically with this command:

```
\fontspec[SmallCapsFont={Art Deco Small Caps}]{Art Deco}
```

ExternalLocation

You can use a font located anywhere on your system, not just in Windows's \Fonts folder. However, automatic selection of bold and italic does not work with external fonts, but you can associate them manually by using the commands given above. Note that you must use the name of the font file, not the display name. Here is an example that points to a Garamond font (gara.ttf) stored in a subfolder called TrueType in a folder Type Outlines:

```
\fontspec[ExternalLocation=\Type Outlines\TrueType]{gara.ttf}
```

Scale

If you have ever mixed, e.g., Times New Roman and Arial in a document, you noticed that the portions in Arial seemed too big even though the same point size was used for both. The `Scale` command can handle this for you. It scales the font being called relative to the default roman font. You can specify an exact percentage for scaling, but usually it's easier to let fontspec adjust it for you as follows:

```
\newfontfamily[Scale=MatchLowercase]{Arial}
```

This scales Arial to match the lowercase characters in Times. If your text is all uppercase, you can use MatchUppercase instead. This feature is particularly useful if you put it in your preamble when defining the default sans-serif and monspaced fonts, which will then work well with the default roman.

Color (or Colour)

The easiest way to get colored text is with the `\textcolor` command. However, fontspec provides another method, as follows, by specifying three two-digit hexadecimal RGB values. The fontspec manual mentions using two additional hex digits to specify the amount of transparency, but this works only on the Mac when using AAT fonts. This example will set some text to green:

```
{\addfontfeature{Color=00BB33}It ain't easy bein' green.---Kermit}
```

It ain't easy bein' green.—Kermit

Note the curly brackets enclosing the entire command, since I didn't want the rest of this document to be green (however much that might make Kermit happy . . .)

LetterSpace

Use this command if you want more spacing between letters than is built into the font, as is commonly done when headings are set in all capitals. (Note that some OpenType fonts can use the Kerning feature with the Uppercase option to handle this also.) The value of 0.0 is the font's default spacing, while a value of 1.0 adds one-tenth of the font's point size to the tracking. So this command:

```
\addfontfeature{LetterSpace=1.0}
```

will add 0.12 points between words if a 12 point font is in effect.

B.5 Font-Dependent Features

These features are tabulated in Appendix A, so they are not repeated here. They will not work unless the font contains the appropriate feature tables.

B.6 Yet More Commands

The following commands are rarely used and so are not discussed here. See *fontspec.pdf* for details.

ADD THESE

ADD HOW TO GET SPECIFIC POINT SIZES

ADD THE MACRO TO ID FEATURES IN A FONT

C Appendix: Some Sample Code

C.1 A Basic Document

To create a simple document, copy the following text, start a new document in TeXworks (or whatever editor you use), and paste from the clipboard. Be sure to copy everything up to, and including, `\end{document}`, which will be on the next page.

```
% !TEX TS-program = xelatex
% !TEX encoding = UTF-8

% this template is specifically designed to be typeset with XeLaTeX;
% it will not work with other engines, such as pdfLaTeX

\documentclass[11pt,letterpaper]{article} % use larger type; default would be 10pt
% this document is based on the ”article” class
% change ”letterpaper” to ”a4” if you use that size paper

% here are packages needed to work with XeTeX

\usepackage{xltxtra} % Extra customizations for XeLaTeX;
% xltxtra automatically loads fontspec and xunicode, both of which you need

% font selection commands using fontspec;
% change the names of the fonts to the ones you want to use!
\defaultfontfeatures{Mapping=tex-text,Scale=MatchLowercase}  % to support TeX conventions like ‘‘--
\setmainfont{Linux Libertine}
\setsansfont{DejaVu Sans}
\setmonofont{DejaVu Sans Mono}

% listed here are some commonly used packages; remove the percentage sign
% at the left margin if you want to load any of them
%\usepackage{color}   % if you want colored text in your document
%\usepackage{url}    % help you typeset URLs
%\usepackage{graphicx} % support the \includegraphics command and options

% for multilingual work you may need polyglossia;
% change the default language and the others to the ones you want
%\usepackage{polyglossia}
%\setdefaultlanguage[variant=american]{english}
%\setotherlanguages{latin,hebrew}

\title{put your title here}
\author{author goes here}
```

```
%\date{} % Activate to display a given date or no date (if empty),
%otherwise the current date is printed

\begin{document}
\maketitle

\section{title of first section}
Text goes here.

\subsection {title of first subsection}
More text goes here.

\end{document}
```

C.2 A Multilingual Sample

Download the zip file containing the sample code. Study the file short multilingual sample.tex, which includes some explanatory comments to help you see what is happening. If you typeset this file, you should see the following as your output if you have the Linux Libertine, DejaVu Sans, and DejaVu Sans Mono fonts on your system. If you don't have them and don't wish to install them, you will need to edit the font definitions in the preamble.

Polytonic Greek
The default language of this document is American English. To get some text in ancient Greek with hyphenation and numerals enabled, we put it inside an environment. So this code:

```
\begin{greek}
Ἡροδότου Ἁλικαρνησσέος ἱστορίης ἀπόδεξις ἥδε, ὡς μήτε τὰ γενόμενα
ἐξ ἀνθρώπων τῷ χρόνῳ ἐξίτηλα γένηται, μήτε ἔργα μεγάλα τε καὶ
θωμαστά, τὰ μὲν Ἕλλησι τὰ δὲ βαρβάροισι ἀποδεχθέντα, ἀκλεᾶ γένηται,
τά τε ἄλλα καὶ δι᾽ ἣν αἰτίην ἐπολέμησαν ἀλλήλοισι.

\medskip
24 = \greeknumber{24}, 1836 = \Greeknumber{1836}

5 = \atticnumeral{5}, 50 = \atticnumeral{50}, 500 = \atticnum{500},
5000 = \atticnum{5000}
\end{greek}
```

produces this output:

Ἡροδότου Ἁλικαρνησσέος ἱστορίης ἀπόδεξις ἥδε, ὡς μήτε τὰ γενόμενα ἐξ ἀνθρώπων τῷ χρόνῳ ἐξίτηλα γένηται, μήτε ἔργα μεγάλα τε καὶ θωμαστά, τὰ μὲν Ἕλλησι τὰ δὲ βαρβάροισι ἀποδεχθέντα, ἀκλεᾶ γένηται, τά τε ἄλλα καὶ δι᾽ ἣν αἰτίην ἐπολέμησαν ἀλλήλοισι.

24 = κδʹ, 1836 = ͵ΑΩΛϚʹ

5 = Π, 50 = 𐅄, 500 = 𐅅, 500 = 𐅆

If you want the acrophonic numerals to work, make sure that you are using a font that supports them.

Latin
Quo usque tandem abutere, Catilina, patientia nostra? quam diu etiam furor iste tuus nos eludet? quem ad finem sese effrenata iactabit audacia? nihilne te nocturnum praesidium Palati, nihil urbis vigiliae, nihil timor populi, nihil concursus bonorum omnium, nihil hic munitissimus habendi senatus locus, nihil horum ora voltusque moverunt? patere tua consilia non sentis, constrictam

iam horum omnium scientia teneri coniurationem tuam non vides? quid proxima, quid superiore nocte egeris, ubi fueris, quos convocaveris, quid consili ceperis quem nostrum ignorare arbitraris? O tempora, o mores! senatus haec intellegit, consul videt; hic tamen vivit. vivit? immo vero etiam in senatum venit, fit publici consili particeps, notat et designat oculis ad caedem unum quemque nostrum. nos autem fortes viri satis facere rei publicae videmur, si istius furorem ac tela vitamus. ad mortem te, Catilina, duci iussu consulis iam pridem oportebat, in te conferri pestem quam tu in nos omnis iam diu machinaris.

Hebrew

א בְּרֵאשִׁית בָּרָא אֱלֹהִים אֵת הַשָּׁמַיִם וְאֵת הָאָרֶץ׃ ב וְהָאָרֶץ הָיְתָה תֹהוּ וָבֹהוּ וְחֹשֶׁךְ עַל־פְּנֵי
תְהוֹם וְרוּחַ אֱלֹהִים מְרַחֶפֶת עַל־פְּנֵי הַמָּיִם׃ ג וַיֹּאמֶר אֱלֹהִים יְהִי אוֹר וַיְהִי־אוֹר׃ ד וַיַּרְא אֱלֹהִים
אֶת־הָאוֹר כִּי־טוֹב וַיַּבְדֵּל אֱלֹהִים בֵּין הָאוֹר וּבֵין הַחֹשֶׁךְ׃ ה וַיִּקְרָא אֱלֹהִים לָאוֹר יוֹם וְלַחֹשֶׁךְ קָרָא
לָיְלָה וַיְהִי־עֶרֶב וַיְהִי־בֹקֶר יוֹם אֶחָד׃ פ ו וַיֹּאמֶר אֱלֹהִים יְהִי רָקִיעַ בְּתוֹךְ הַמָּיִם וִיהִי מַבְדִּיל בֵּין
מַיִם לָמָיִם׃ ז וַיַּעַשׂ אֱלֹהִים אֶת־הָרָקִיעַ וַיַּבְדֵּל בֵּין הַמַּיִם אֲשֶׁר מִתַּחַת לָרָקִיעַ וּבֵין הַמַּיִם אֲשֶׁר
מֵעַל לָרָקִיעַ וַיְהִי־כֵן׃ ח

We can mix English and Hebrew, as in a commentary:

The first three words בְּרֵאשִׁית בָּרָא אֱלֹהִים are analyzed as follows…

Old Italic

SAMPLE AND DISCUSSION GO HERE

D Appendix: Fonts with OpenType Features

Now that you know how to use OpenType features, you will want to experiment. This table includes information about freely available fonts to try. There are of course commercial fonts, particularly those from Adobe.

FONT	DESCRIPTION AND URL
Cardo	Based on Renaissance designs, Cardo is designed for use by scholars in classics, biblical studies, medieval studies, linguistics, and related fields. It contains a very large glyph complement but only comes in regular weight (for now). A new version is in preparation (September 2010) that will contain many OT features for advanced typography. http://scholarsfonts.net
Charis SIL	Based on the highly legible Bitstream Charter design, Charis SIL is a family with four faces and supports Latin- and Cyrillic-based languages along with the special needs of linguists. It contains feature tables for AAT, OT, and Graphite. Charis SIL handles placement of combining diacritics better than almost any other font. It also contains small caps and some language-specific features (Vietnamese, Romanian, etc.). See : http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=CharisSILFont
Gentium Basic	A font family with four faces, Gentium Basic supports Latin-script fonts only. It uses OT and Graphite features to handle placement of diacritics and alternate forms needed for various languages. Note that the character repertoire is much smaller than the main Gentium family. Gentium Book, with slightly heavier outlines, is also available in four faces, with the same characters as Gentium Basic. http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=Gentium_basic
GFS	The Greek Font Society makes available a number of high-quality typefaces, some of which include support for Latin-script languages as well as Greek and include a variety of OT features. Particularly notewprthy is GFS Neohellenic, which includes (via the OT stylistic alternates feature) the alternate letterforms that Victor Scholderer designed in 1927 but which have not been available in most versions of ths popular typeface. See www.greekfontsociety.gr

FONT	DESCRIPTION AND URL
Junicode	Designed for medievalists, Junicode contains a large character repertoire and can be used for many languages. It also handles combining marks well and contains many OT features. The bold and italic contain fewer characters than the roman. http://junicode.sourceforge.net/
Linux Libertine	Despite its name, this font runs on Windows and Mac OS just fine. It is a Renaissance-style face in four weights with many OT features. Linux Biolinum, a sans-serif face, is under development. See http://linuxlibertine.sourceforge.net/
TeX Gyre	TeX Gyre is a project to update and extend the fonts distributed with the open-source Ghostscript page description language. It includes a number of fonts, each in OpenType and Type 1 formats. The OT versions contain many features for advanced typography, all of which are identified in the documentation. Except as noted, all families include four fonts. The families released so far are: Adventor: a sans-serif face similar to Avant Garde Gothic Bonum: a serif face similar to Bookman Old Style Chorus: an italic-only face similar to Zaph Chancery Cursor: a monospaced font similar to Courier Heros: a sans-serif face similar to Helvetica; both standard and condensed versions of each font included (8 fonts) Pagella: a serif face similar to Palatino Schola: a serif face similar to Century Schoolbook Termes: a serif font similar to Times http://www.gust.org.pl/projects/e-foundry/tex-gyre/

E Appendix: Some Traditional TEX Keystrokes

See page 19 for background on these keystrokes.

ACCENT	TYPED AS	RESULT
macron	\={o}	ō
breve	\u{o}	ŏ
overdot	\.{o}	ȯ
underdot	\d{o}	ọ
line below	\b{o}	o̲
tie	\t{oo}	ôo
acute	\’{o}	ó
grave	\‘{o}	ò
circumflex	\^{o}	ô
umlaut/tréma	\”{o}	ö
double acute	\H{o}	ő
caron	\v{o}	ǒ
cedilla	\c{c}	ç
ogonek	\k{o}	ǫ

CHARACTER	TYPED AS	RESULT
dotless i	\i	ı
dotless j	\j	ȷ
A/a-ring	\AA, \aa	Å, å
AE/ae ligature	\AE, \ae	Æ, æ
L/l-slash	\L, \l	Ł, ł
Eth/eth	\DH, \dh	Ð, ð
	\DJ, \dj	Đ, đ
Eng/eng	\NG, \ng	Ŋ, ŋ
OE/oe ligature	\OE, \oe	Œ, œ
O/o-slash	\O, \o	Ø, ø
sharp s	\SS,\ss	SS, ß
Thorn/thorn	\TH, \th	Þ, þ

SYMBOL	TYPED AS	RESULT
en-dash	\--	-
em-dash	\---	–
underscore	_	_
inverted marks	?‘, !‘	¿, ¡
copyright	\copyright	©
dagger	\dag	†
double dagger	\ddag	‡
paragraph	\P	¶
pound sign	\pounds	£
section	\S	§

Change Log

1.6	Sept. 9, 2010	Many small changes and corrections; some items rewritten for clarity, particularly in the section on RTL text; URLs updated; changes to reflect new versions of fontspec and bidi.
1.5	June 28, 2009	Hyperlinks added using hyperref; more resources for Mac and Linux users added; sample code revised; this version never actually posted on the web.
1.4	June 20, 2009	Section on RTL text rewritten and expanded, based on recent testing and emails (now complete); section on creating multilingual text expanded and improved, with examples and info on xunicode; Appendix E (traditional TeX keystrokes) added.
1.3	May 24, 2009	Discussion of the different forms of font controls (end of §7.1) added; discussion of \newfontfamily and \newfontface clarified; miscellaneous typos fixed and some formatting improvements.
1.2	May 18, 2009	Sections on fontspec and polyglossia significantly enlarged and improved; information on hyphenation added; note to Mac and Linux users added; sample code added; miscellaneous improvements.
1.0	May 10, 2009	Initial release.