Arabic font building for LaTeX

F. Mounayerji, M. A. Naal
Department of Computer Engineering
University of Aleppo, Syria
Fares_Mounayerji (at) hotmail dot com

Abstract

This contribution aims to describe a new solution for building arabic font for LaTeX. We focus on the font generation for Arabic calligraphy. This solution is based on the determination of control points that gives precise METAFONT code for the given Arabic font glyph. Using the METAFONT compiler, the new font is compiled and finally installed on a LaTeX distribution.

1 Introduction

The Arabic language is one of the ten most commonly used languages worldwide. Over 300,000,000 people living in the Arab world use this language in everyday and official writing. This creates an important potential for Arabic text editor users.

LaTeX is an elegant and advantageous typesetting program and is strongly recommended for scientific writing. Its capabilities and especially its mathematical capabilities are well known.

LaTeX uses a logical structure or WYMIWYG (What You Mean Is What You Get) concept instead of WYSIWYG (What You See Is What You Get), which makes LaTeX unique in its approach for building texts, and building structure-oriented rather than formatting-oriented text, which reduces errors and increases concentration on the idea of the text.

There is a growing interest in globalizing LaTeX by supporting the most important languages all over the world, and as we know, the Arabic language is important and widely used, which generates a need for even more extensive support than that of the existing ArabTeX and other Arabic support packages. We also need to give the Arabic user of LaTeX more options, such as font selection, font adding, special Arabic effects, etc.

Because of all the previous factors mentioned, we see the importance to expand the support for this important language.

2 LaTeX Principles

2.1 What is LaTeX?

LaTeX is a programming language used to build large text documents (e.g., books, articles, theses) presentations, etc. We can also use the term LaTeX for a LaTeX compiler. Figure 1 shows a simplified view of the structure of LaTeX processing.

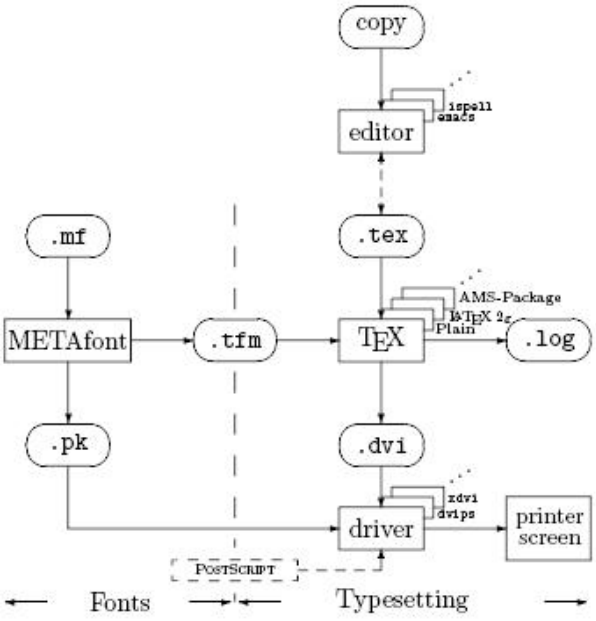

Figure 1: Simplified LaTeX system structure

2.2 What is a LaTeX distribution?

A (LA)TeX distribution is a set of folders and files containing a (LA)TeX compiler, in addition to supporting tools such as a previewer, METAFONT compiler, and much more. Some LaTeX distribution is needed to run LaTeX on a given machine; optionally, a front-end like TeXnicCenter or WinEdt can also be used. LaTeX is free and open source software, which has led to an enormous number of distributions, but only a few of them are recognized worldwide, such as MiKTeX on Microsoft Windows,

MacTeX on Mac OS X, and teTeX which is usually installed by default on Linux platforms.

Standard distributions follow the TDS (TeX Directory Structure) conventions, which describe how to organize a distribution's files and folders, and in relation to our main subject of font files, the TDS specifies where to put the font files such as `.pk`, `.tfm` and `.mf`.

2.3 METAFONT

METAFONT is a font building programing language used to build fonts for (LA)TeX. It has a special syntax to do this, depending on the control points of the letter and their ordering. A line is drawn passing between the control points, sometimes a straight line, and sometimes a curved line [5].

3 The solution

Because of the importance of the Arabic language, Arabic fonts are important too. Every user should have the ability to design the font that he wants to use with his distribution of LaTeX. To achieve this goal we built a solution that permits any character, or actually any shape, to be processed in different stages in order to achieve the font as the user wants it to be. The main stages in achieving this goal are:

- drawing the font on paper,
- scanning the font as an image,
- analyzing the image,
- finding the relations between control points,
- and generating the METAFONTcode.

We will talk about each of these stages in detail in the following sections.

3.1 Drawing the font on paper

This step is drawing the Arabic character like و, ص or any other characters on paper, respecting Arabic calligraphy. We don't need to draw the double-struck letter forms because there is an algorithm to extract it from the standard form.[1]

3.2 Scanning the font

This will be a simple procedure of scanning the font in order to enter it into the computer machine, and deal with it as a simple colored image.[2] We also include applying some changes to the image to be ready for the next step.

[1] It is worth mentioning that we can build our font as a dynamic font [2] by using the kashida, a curvilinear variable lengthening of letters along the baseline.

[2] Usually black and white, with the font being black and the background white.

First stage Transform the image into a grayscale image, and then into a black and white image.

Next stage Applying certain algorithms in order to prepare the image for determining the control points, such as the contour algorithm for the double-struck letter, and the skeleton algorithm for the standard form, as we can see in Figure 2.

Figure 2: Standard, skeleton and contour

3.3 Analyzing the image

This step will deal with the output image of the previous step, and it will be responsible for two main objectives:

- Determining the control points of the letter as precisely as possible. Since these points will be used later to build the METAFONT code, this procedure is the main work here. We will scan the image line by line until we find a black point; this point will be the start of the letter (generally but not always — the font could be composed of lines and curves, separated or connected). After that, we will move on those curves and lines point by point, and during this pass we will save the information that we get about them in matrices, in order to be used in a later stage for building the font as it appears in the image.
- Determining the path between those points is very important here because we will lose the original shape of the letter without knowing the exact path between its control points.[3]

Figure 3 shows how we handle the standard case by using the skeleton algorithm, and Figure 4 shows how we handle the contour case or the double-struck character.

3.4 Finding the relations between the control points

This is an important step for simplifying the METAFONT code, because many relations between the control points could be found. Simplifying will make the

[3] The control points cannot be repeated but the path points can be repeated.

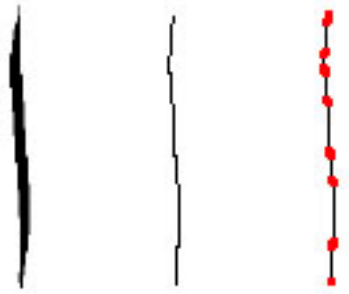

Figure 3: Standard, skeleton and control points

Figure 4: Double-struck and control points

work for building the METAFONT file easier, and the result more professional. Those relations can vary from two points or more which have the same x or y, to two points have a point which fall between them in three cases:

- $x_3 = (x_1 + x_2)/2$
- $y_3 = (y_1 + y_2)/2$
- $z_3 = (z_1 + z_2)/2$

3.5 Generating the METAFONT code

This step will include transforming the results of the previous steps into code readable by the METAFONT compiler. First of all we need to transform the Cartesian coordinates of the solution environment into the Cartesian coordinates of the METAFONTcompiler. Secondly we will have to transform the points of the path into the METAFONT form. And finally add the METAFONT instructions that will draw the lines between the previous points.

3.6 Installing the new font

This step actually depends on the particular (LA)TEX distribution, but we can specify the main stages in a general way [1]. Assuming that the METAFONT file is `arab.mf`:

- From the command line, run

```
mf '\mode=ljfour; mode_setup; input arab.mf'
```

- The output files will be `arab.tfm` and `arab.600gf` (or similar).[4]
- Now you still need to pack the `gf` file into a usable format. You do so with this command:

```
gftopk arab.600gf arab.pk
```

- And that's it! Your font is now available to every LATEX document whose source is in the same directory as the font files. I mean here both file with the extensions `.pk` and `.tfm`. We can also install it permanently, which is a process that depends on the distribution [3].

References

[1] Christophe Grandsire, *The METAFONT tutorial*, Version 0.33.

[2] Mostafa Banouni, Mohamed Elyaakoubi and Azzeddine Lazrek, Dynamic Arabic Mathematical Fonts. *Preprints for the 2004 Annual Meeting*, Xanthi, Greece, pp. 48–53, 2004.

[3] TUG Working Group on a TEX Directory Structure (TWG-TDS), *A Directory Structure for TEX Files*, version 1.1, June 23, 2004.

[4] Azzeddine Lazrek, *NasX Arabic literal symbols font*, May 2, 2004.

[5] Peter Flynn, *Formatting information: A beginner's introduction to typesetting with LATEX*, 2005.

[6] *The UK TEX FAQ, Your 396 Questions Answered*, version 3.15a, date 2005/11/29.

[7] Jeff Clark, *LATEX Tutorial*, revised February 26, 2002.

[8] Klaus Lagally, *ArabTEX, Typesetting Arabic and Hebrew, User Manual*, Version 4.002, March 11, 2004.

[4] The number 600 indicates that the precision is 600 dpi (dots per inch).